



Eric Rocha de Souza

BSc in Information Systems
Specialist in Software Engineering
MSc in Computer Engineering

A Value-Driven Framework for Software Architecture

Thesis submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy in
Computer Science

Adviser: Ana Maria Diniz Moreira,
Associate Professor with Habilitation,
NOVA University of Lisbon

Examination Committee

Chairperson: Professor Luís Manuel Marques da Costa Caires
Raporteurs: Professor Jaelson Brelaz de Castro
Professor António Rito Silva
Members: Professor Luís Manuel Marques da Costa Caires
Professor Ana Maria Diniz Moreira
Professor Sílvia Mara Abrahão Gonzales
Professor Ademar Manuel Teixeira de Aguiar
Professor João Baptista da Silva Araújo Júnior

A Value-Driven Framework for Software Architecture

Copyright © Eric Rocha de Souza, Faculty of Sciences and Technology, NOVA University Lisbon.

The Faculty of Sciences and Technology and the NOVA University Lisbon have the right, perpetual and without geographical boundaries, to file and publish this dissertation through printed copies reproduced on paper or on digital form, or by any other means known or that may be invented, and to disseminate through scientific repositories and admit its copying and distribution for non-commercial, educational or research purposes, as long as credit is given to the author and editor.

I dedicate this work to my wife Renata.

ACKNOWLEDGEMENTS

Obtaining a Ph.D. degree is a demanding journey, which is only possible with the help of those who are closer to us, even when they are geographically distant. I would like to thank all those who have supported me in this mission. First and foremost, none of this would have been possible without the help of my friend and mentor Professor Ana Moreira. It was really a privilege to work under her supervision. Thank you for your friendship, availability, time, attention, patience, encouragement, lessons, and all support during this period. My thanks also to all those who have helped me day after day during these years. There are too many to list, but I can not forget all the assistance given to me by the professors of the Doctoral Program in Computer Science from the Departamento de Informática of Universidade NOVA de Lisboa. Thank you all professors and researchers from the Automated Software Engineering research group, in particular Professor João Araújo, for his support and guidance, and my colleagues Fernando Wanderley and Cristiano de Faveri for all their suggestions, critiques, and the constant joy of living that made the distance from my homeland shorter. Thank you also to all participants of the empirical studies, especially to Silvia Abrahão and Emilio Insfran for the receptivity and assistance during my stay in Valencia, where I started the evaluation of my Ph.D.. Many thanks to the Master's students in Computer Science from Universitat Politècnica de València (Spain), the Master's students in Computer Science and Informatics from Universidade NOVA de Lisboa (Portugal), and the Doctoral students in Business Management from Universidade Federal de Pernambuco (Brazil) for their willingness to participate in the empirical experiments that are part of the evaluation of this research work. My sincere gratitude to the institutions that contributed in some way to the success of this work, particularly Departamento de Informática from Universidade NOVA de Lisboa, NOVA LINCS research Lab (Ref. UID/CEC/04516/2013), Value@Cloud project (MINECOTIN2013-46300-R), Department of Computer Systems and Computation from Universitat Politècnica de València, and Programa Ciência sem Fronteiras from Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – CAPES (Ref. 99999.009047/2013-01). Finally, last but not the least, I would like to thank my wife, Renata, who left everything behind in Brazil to live my dream with me in Portugal. Thank you also to my beloved family. Much of the time spent on this job was directly stolen from you. This thesis is also yours.

*“How wonderful it is that nobody need wait a single moment
before starting to improve the world.” – Anne Frank*

ABSTRACT

Software that is not aligned with the business values of the organization for which it was developed does not entirely fulfill its *raison d'être*. Business values represent what is important in a company, or organization, and should influence the overall software system behavior, contributing to the overall success of the organization. However, approaches to derive a software architecture considering the business values exchanged between an organization and its market players are lacking. Our quest is to address this problem and investigate how to derive value-centered architectural models systematically. We used the Technology Research method to address this PhD research question. This methodological approach proposes three steps: problem analysis, innovation, and validation. The problem analysis was performed using systematic studies of the literature to obtain full coverage on the main themes of this work, particularly, business value modeling, software architecture methods, and software architecture derivation methods. Next, the innovation step was accomplished by creating a framework for the derivation of a software reference architecture model considering an organization's business values. The resulting framework is composed of three core modules: Business Value Modeling, Agile Reference Architecture Modeling, and Goal-Driven SOA Architecture Modeling. While the Business value modeling module focuses on building a stakeholder-centric business specification, the Agile Reference Architecture Modeling and the Goal-Driven SOA Architecture Modeling modules concentrate on generating a software reference architecture aligned with the business value specification. Finally, the validation part of our framework is achieved through proof-of-concept prototypes for three new domain specific languages, case studies, and quasi-experiments, including a family of controlled experiments. The findings from our research show that the complexity and lack of rigor in the existing approaches to represent business values can be addressed by an early requirements specification method that represents the value exchanges of a business. Also, by using sophisticated model-driven engineering techniques (e.g., metamodels, model transformations, and model transformation languages), it was possible to obtain source generators to derive a software architecture model based on early requirements value models, while assuring traceability throughout the architectural derivation process.

In conclusion, despite using sophisticated techniques, the derivation process of a software reference architecture is helped by simple to use methods supported by black box transformations and guidelines that facilitate the activities for the less experienced software architects. The experimental validation process used confirmed that our framework is feasible and perceived as easy to use and useful, also indicating that the participants of the experiments intend to use it in the future.

Keywords: Software and its engineering; *Software architectures; Model-driven software engineering; Domain specific languages; Requirements engineering; Software development methods; Value-driven development; Business modeling.*

RESUMO

Software que não está alinhado com os valores comerciais da organização para a qual foi desenvolvido não cumpre inteiramente sua missão. Os valores de negócio representam o que é importante numa empresa ou organização e por isso devem influenciar o comportamento geral do sistema de software, contribuindo para o sucesso geral da organização. Todavia, falta uma abordagem para derivar uma arquitetura de software considerando os valores de negócio trocados entre uma organização e seus parceiros de mercado. O nosso objetivo é endereçar este problema, investigando como obter sistematicamente modelos arquiteturais centrados em valor. Para isso, usamos o método de Pesquisa Tecnológica (do inglês, *Technology Research method*) para resolver a pergunta de investigação do nosso trabalho de doutoramento. Esta abordagem metodológica propõe três etapas: análise do problema, inovação e validação. A análise do problema foi realizada à custa de estudos sistemáticos da literatura para obter uma cobertura completa dos principais temas deste trabalho, particularmente modelação de valores de negócio, métodos de arquitetura de software e métodos de derivação de arquitetura de software. Em seguida, a etapa de inovação foi realizada com a criação de um *framework* para a derivação de um modelo de arquitetura de referência de software, considerando os valores de negócio de uma organização. O *framework* resultante é composto por três módulos principais: Modelagem Modelação de Valor de Negócio, Modelação de Arquitetura de Referência Ágil e Modelação de Arquitetura SOA Orientada por Objetivos. Enquanto o módulo de modelação de valor de negócio se concentra na construção de uma especificação de negócio, os módulos modelação de arquitetura de referência ágil e modelação de arquitetura SOA orientada por objetivos concentram-se na geração de uma arquitetura de software de referência alinhada com a especificação de valor. Finalmente, a parte da validação foi alcançada por meio de protótipos de prova de conceito para três novas linguagens de domínio específico, estudos de caso e quase-experimentos, incluindo uma família de experimentos controlados. As descobertas do nosso trabalho de investigação mostram que a complexidade e a falta de rigor nas abordagens existentes para representar valores de negócio podem ser tratadas por um método de especificação de requisitos logo nas fases iniciais do desenvolvimento que representa as trocas de valores de um negócio. Adicionalmente,

usando técnicas sofisticadas de engenharia orientada a modelos (por exemplo, metamodelos, transformações de modelos e linguagens de transformação de modelos), foi possível obter geradores para derivar um modelo de arquitetura de software com base em modelos de requisitos de valor, garantindo a rastreabilidade em todo o processo de derivação arquitetural.

Palavras-chave: Software e sua engenharia: *Arquitetura de software; Engenharia de software dirigida por modelos; Linguagens específicas de domínio; Engenharia de requisitos; Métodos de desenvolvimento de software; Desenvolvimento dirigido por valor; Modelação de negócio.*

CONTENTS

List of Figures	xix
List of Tables	xxiii
Listings	xxv
Glossary	xxvii
Acronyms	xxxiii
1 Introduction	1
1.1 Context and motivation	1
1.2 Problem statement	2
1.3 Challenges	4
1.4 Research questions	6
1.5 Supporting methodologies, paradigms, and technologies	7
1.6 Major results	11
1.7 Research methodology	14
1.8 Structure of this document	15
2 Business modeling	19
2.1 Overview on business modeling	19
2.1.1 Value-based software engineering	19
2.1.2 Value-driven modeling	21
2.2 A mapping study on business models	23
2.2.1 Planning: research protocol	24
2.2.2 Conducting: search results	27
2.2.3 Reporting: answering the research questions	29
2.2.4 Threats to validity	32
2.3 Two Business modeling approaches	33
2.3.1 Eriksson-Penker UML business extensions	33
2.3.2 e3value	34
2.4 Final considerations	37

3	Software architecture	39
3.1	What is software architecture?	39
3.2	Service-oriented architecture	43
3.3	State of the art on software architecture: an Evidence-Based Tertiary Study	47
3.3.1	Planning: Defining the protocol	48
3.3.2	Conduction	54
3.3.3	Reporting: Answering the research questions	61
3.3.4	Threats to validity	70
3.4	Final considerations	71
4	Deriving architectural models from requirements specifications	73
4.1	Planning	73
4.1.1	Formulating the research questions	74
4.1.2	Formulating the search string	74
4.1.3	Defining the search strategies	75
4.1.4	Selecting the research sources	75
4.1.5	Selecting studies	75
4.1.6	Assessing the quality of the studies	76
4.1.7	Collecting the Data	78
4.1.8	Reviewing the protocol	79
4.2	Conduction	81
4.3	Reporting: Study results	84
4.3.1	Demographic data	84
4.3.2	Context	84
4.3.3	Benefits to the users	88
4.3.4	Content	88
4.3.5	Validation	98
4.4	Overview of the results	99
4.4.1	Context	99
4.4.2	Benefit to the users	99
4.4.3	Content	99
4.4.4	Validation	100
4.5	Validity threats and their mitigation	100
4.6	Research roadmap	101
4.7	Final considerations	103
5	A value-driven framework for software architecture	105
5.1	Framework's structure	105
5.2	Business value modeling	107
5.2.1	DVD in a nutshell	107
5.2.2	DVD abstract syntax and constraints	109

5.2.3	DVD concrete syntax	112
5.2.4	DVD process	113
5.2.5	From a DVD model to a SoaML capability model	115
5.2.6	About the DVD implementation	117
5.3	Agile reference architecture modeling	118
5.3.1	An overview of the supported agile concepts	118
5.3.2	RAMA in a nutshell	120
5.3.3	RAMA abstract syntax and constraints	121
5.3.4	RAMA concrete syntax	127
5.3.5	The RAMA process	128
5.3.6	About the RAMA implementation	133
5.4	Goal-driven SOA architecture modeling	133
5.4.1	Goal-oriented approaches	134
5.4.2	KAOS4Services in a nutshell	136
5.4.3	KAOS4Services abstract syntax, constraints, and concrete syntax	137
5.4.4	KAOS4Services process	139
5.4.5	About the KAOS4Services implementation	146
5.5	Final considerations	146
6	Case Study	147
6.1	Business description	147
6.2	Applying the DVD method	148
6.3	Applying RAMA method	150
6.4	Applying KAOS4Services method	158
6.5	Final considerations	165
7	Evaluation through experiments	167
7.1	Evaluating the DVD method	168
7.1.1	Experiment design	168
7.1.2	Discussion of the quasi-experiment results	170
7.1.3	Threats to validity for the quasi-experiment	170
7.2	Comparing the methods DVD and e3value	171
7.2.1	Experimental methodology	172
7.2.2	Baseline Experiment	173
7.2.3	Discussion of results	180
7.2.4	Two Experimental Replications	182
7.2.5	Meta-Analysis	185
7.2.6	Discussion of the results	186
7.2.7	Threats to validity for the family of the controlled experiments	190
7.3	Evaluating the methods RAMA and KAOS4Services	192
7.3.1	Experiment design	192

CONTENTS

7.3.2	Discussion of the evaluation results for RAMA and KAOS4Services	194
7.3.3	Threats to validity	196
7.4	Comparing RAMA and KAOS4Services	196
7.4.1	Experiment design	197
7.4.2	Complementary analysis	199
7.4.3	Discussion of the results	200
7.4.4	Threats to validity for the survey	201
7.5	Final considerations	202
8	Conclusions and future work	203
8.1	The quest and respective research questions	203
8.2	Contributions	206
8.3	Published results	209
8.4	Future work	210
	Bibliography	213

LIST OF FIGURES

1.1	Relating problem statement to challenges.	6
1.2	Transformation scheme [137].	10
1.3	Document structure.	16
2.1	A company that offers its product to a consumer using a content provider, a producer (or production house), a network infrastructure provider, and some carriers (e.g., a telecom operator). It uses a network integrator (layer above) and facilitates the management of operations as support (layer below) — details in [38].	23
2.2	Cisco outsources the manufacturing of its products through Component Suppliers (who produce the electronic components), Manufacturers/Assemblers (who produce the Cisco's products), and distributors (who transports the electronic components from the Component Suppliers to the Manufacturers/Assemblers). With the products in hand, Cisco sells them using Sales Channels (electronics stores) or directly to the Customers (details in [250]).	24
2.3	Planning phase.	25
2.4	EPBE metamodel.	34
2.5	e3value metamodel, taken from [104].	36
2.6	e3value example extracted from [206].	36
3.1	General view of a Service-Oriented Architecture [235].	45
3.2	Example of a SoaML capability model for an order handling system.	46
3.3	a) Ecore SoaML ServicesArchitecture metamodel from [189] extended with part of the QoS& FT profile, and b) SoaML specification, taken from [77].	47
3.4	Study searching and selection process.	54
3.5	Publication period of the selected systematic studies.	59
3.6	Number of citations per secondary study.	70
4.1	Distribution of Non-functional Requirements (NFRs) (left) and requirements models (right).	87
4.2	Distributions of architectural views as reported in the primary studies (left) and mapped to the 4+1 views (right).	97

5.1	Modules, environments, and technologies of value-driven framework.	106
5.2	Illustration of the DVD concepts.	109
5.3	Dynamic Value Description metamodel.	110
5.4	Concrete notation for the DVD language.	112
5.5	Example of a DVD model for the purchase and sale of goods by a store expressed using the DVD concrete syntax.	112
5.6	Process to create a DVD model.	113
5.7	SoaML capability model generated from a DVD model.	117
5.8	Subway map to agile practice [9].	119
5.9	User story metamodel.	122
5.10	User story sketch.	123
5.11	Conceptual model metamodel.	124
5.12	Conceptual model sketch.	125
5.13	Metamodel for traceability support.	126
5.14	Traceability (tree) model sketch.	127
5.15	User story specification with the concrete syntax.	128
5.16	Traceability tree model concrete syntax	129
5.17	RAMA's process.	129
5.18	KAOS structure example from [237].	136
5.19	KAOS4Services metamodel.	138
5.20	A Goal needs a relation.	139
5.21	KAOS4Services process.	139
5.22	Applying the guidelines 3.1–3.5	143
5.23	Candidate services, using guidelines 4.1–4.5. The colored numbers indicate the order of the operations or activities.	145
5.24	Applying guidelines 6.1 and 6.2	146
6.1	(a) DVD editor palette and (b) the value exchange properties.	148
6.2	DVD model created using the tool.	149
6.3	Changing the focus of analysis.	150
6.4	Menu to open the transformation execution window.	150
6.5	Execution window.	151
6.6	The capability model generated.	151
6.7	“Registering data” user story.	153
6.8	Prioritization of value exchanges.	154
6.9	Tree model generated after prioritization. Note that the value exchanges are represented grouped in High (red), Medium (yellow), and Low (blue) priorities according to RoI.	154
6.10	Conceptual model for “Registering data” user story.	155
6.11	“Buying in the gas station” conceptual model.	155
6.12	Conceptual model for “Placing a bet” user story.	156

6.13	Conceptual model for “Buying the auctioned good” user story.	156
6.14	Reference architecture for the online auction system.	157
6.15	Initial KAOS model for the auction online system.	158
6.16	Propeties of “Placing a bet” requirement.	159
6.17	Decomposition of requirement “ <i>Register the personal data</i> ”.	159
6.18	Decomposition of requirement “ <i>Register a bet</i> ”.	160
6.19	Decomposition of requirement “ <i>Make a payment</i> ”.	160
6.20	Decomposition of requirement “ <i>Buy product</i> ”.	161
6.21	Business processes generated from the requirement “user registration”. . . .	162
6.22	Business processes generated from the requirement “user authentication”. .	162
6.23	Business processes generated from the requirement “register a bet”.	162
6.24	Business processes generated from the requirement “make checkout”.	162
6.25	Business processes generated from the requirement “buy product”.	163
6.26	Identifying the candidate services.	163
6.27	Consolidating the candidate services.	164
6.28	Generating the architectural model.	165
6.29	SoaML model for online auction system.	166
7.1	Overview of our family of experiments.	173
7.2	Taxonomy of dependent variables.	174
7.3	Processes for the creation of (a) an e3value model and (b) a DVD model. . . .	176
7.4	Initial oracles of (a) e3value and (b) DVD for Object1.	176
7.5	Analysis procedure employed for the experiment in Spain (UPV).	181
7.6	Analysis procedure employed for the experiment in Portugal (UNL).	183
7.7	Analysis procedure employed for the experiment in Brazil (UFPE).	184
7.8	Meta-analysis blobbogram for effectiveness, efficiency, PEOU, PU and ITU. .	186
7.9	Actual efficacy (effectiveness and efficiency), perceived efficacy (PEOU and PU), and Intention To Use (ITU) grouped by methods of experiments per- formed in (a) Spain, (b) Portugal, and (c) Brazil.	187

LIST OF TABLES

1.1	List of publications.	13
2.1	Application of the Filtering Criteria in business modeling research.	27
2.2	Selected studies for the mapping study on business modeling.	27
2.3	Synthesis of the data extracted on business modeling.	29
3.1	Software architecture definitions.	40
3.2	Population, Intervention, Comparison, Outcome, and Context (PICOC) Analysis for the tertiary study about secondary studies on Software Architecture.	49
3.3	Research Query Building	51
3.4	Search results for automated and manual searches.	54
3.5	Selected studies for the tertiary study on software architecture.	55
3.6	List of selected systematic studies.	59
3.7	Number of primary studies per review, time span of review and if the review shows the primary study list	62
3.8	Categories and corresponding number of secondary studies	65
3.9	Number of reviews and authors per country	69
3.10	Number of reviews per author's affiliation with at least 2 studies.	70
4.1	PICOC Analysis	74
4.2	Research Query Building	75
4.3	Quality Assessment Checklist	77
4.4	Research question decomposition	78
4.5	Evaluation Score	80
4.6	Application of the filtering criteria	81
4.7	Selected studies for the mapping study on software architecture derivation methods.	82
4.8	Context of the selected studies.	85
4.9	Requirements Types and Models Used as Input, where ✓ means requirements type/model used and ✗ means not used.	86
4.10	Coverage of the methods, where ✓ means phase covered and ✗ means phase not covered.	89

4.11 Mapping primary study views to Kruchten 4+1 views; when authors tackle more than one view, we address each sequentially.	90
5.1 Concrete language for User story specification tool.	128
5.2 Goal-oriented concepts	134
5.3 Heuristics (concepts map)	135
6.1 User stories	151
6.2 Overlaps	157
6.3 Applying guidelines 4.1 – 4.5 to identify candidate services.	163
7.1 Experiment design	170
7.2 The descriptive statistics for PEOU and PU	170
7.3 Experiment design	179
7.4 Descriptive statistics for effectiveness, efficiency, PEOU, PU, and ITU per experiment and method.	180
7.5 Summary of the results of all experiments, where checkmark means hypothesis accepted and X means hypothesis rejected.	188
7.6 Experiment design	194
7.7 Effect size of survey	199

LISTINGS

5.1	Example of DVD semantic rule	111
5.2	M2M Transformation: from a DVD model to a SoaML capability model .	116
5.3	Given step must have only link to a Content	123
5.4	Attributes can only link to Entities	125
5.5	A root node has three priorities nodes	127
5.6	Levenshtein Distance method	131
5.7	Rule 1: Create software components	132
5.8	Rule 2: Create associations	132
5.9	Transformation from DVD to KAOS	140

GLOSSARY

Actor	A DVD actor is an economically independent entity. A company, business unit, role, or a customer are examples of actors. “Economically independent” refers to the ability of an actor to be profitable or to increase value for him/herself.
Agile practices	Agile practices aim at reducing the effort-intensive tasks in agile software development, focusing on fast response to the various changes in a project [83].
Agile software development	Agile software development is any software development methodology where specification and solutions evolve through collaboration between self-organizing teams.
Architectural style	Architectural styles determine the set of principles and guidelines used to design an architecture software [172].
Business model	Business model is an efficient way of understanding, evaluating, managing and conveying the core concepts of a business [107, 198]. It can be considered the basis to design an information system that supports the needs of a company [271].
Business modeling	Business modeling is a conceptual tool to express the business logic of an organization, describing the set of values that the organization offers to its clients and the network of its partners with which it exchanges values in a way that is sustainable and cost-effective [192].

Domain Specific Language	A Domain Specific Language (DSL) is a language designed to be useful for a specific set of tasks of a particular domain [109].
Goal-oriented model	A goal-oriented model uses goal as the concept to provide the rationale for the envisioned system [257].
Goal-oriented requirements engineering	Goal-oriented requirements engineering uses goals for eliciting, elaborating, structuring, specifying, analyzing, negotiating, documenting and modifying requirements [257].
Metamodel	A metamodel is a model that defines the language for expressing a model [177].
Model	Model is an abstract representation of a system (or a theory or reality) that allows one to make inferences and predictions about the system [163].
Model-Driven Development	Model-Driven Development uses abstraction and automation through the transformations between models as the two core concepts to develop software systems [177].
Model-Driven Engineering	Model-Driven Engineering is a software development methodology that focuses on the creation and exploration of domain models, ie, conceptual models representing all topics related to a specific problem [177].
Non-functional Requirements	Non-Functional Requirements (NFRs) typically define the overall qualities of a system, as well as the constraints imposed on a solution [58].
Oracle	Oracle is used to refer to the correct model created by an expert. It is used as a baseline to measure the models created by the participants of the evaluation experiments.

Service	A Service is the SOA core concept. Service is defined as an autonomous, minimally coupled, and platform-independent entity that can be published and used in novel ways [195].
Service-Oriented Architecture	Service-Oriented Architecture (SOA) is an architectural style to support the development of distributed applications, even in heterogeneous environments [136, 196].
Software reference architecture	A Reference architecture offers a template solution of an architecture for a particular domain, expressing ways of organizing the fundamental structure of a system through high-level reusable solutions [24, 235].
Software architecture	Software architecture has been defined in many different ways [120], but at its core it refers to the software elements, their relationships with each other and its environment, the principles governing its design and evolution [127], as well as the quality properties these elements support [24].
Software engineering	The systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, testing, and documentation of software to optimize its production, maintenance, evolution, and quality [126].
Value	Value is the reason why companies and people trade with each other, exchanging goods among them [104].
Value level agreement	A value level agreement is a particular business aspect that must be minimally agreed among the actors in order to enable the value exchanges; it defines the business constraints based on the business strategies.

Value object	A value object is a service, a product, or even an experience, which is of economic value for at least one of the actors involved in a value model.
Value port	A value port is used to interconnect actors so that they are able to exchange value objects.
Value exchange	A value exchange represents one trade of value object instances between value ports.
Value-based software engineering theory	Value-based software engineering theory is a research area that combines the traditional computer science theories with value-based theories to provide processes and a framework for guiding Value-Based Software Engineering (VBSE) activities.
Value-based people management	Value-based people management is a research area that includes managing the expectations, as well as the project's accommodation of all stakeholders' value propositions.
Value-based quality management	Value-based quality management is a research area that involves prioritization of desired quality factors concerning the stakeholders' value propositions.
Value-based risk management	Value-based risk management is a research area that combines principles and practices to identify, analyze, prioritize and mitigate risk.
Value-based planning and control	Value-based planning and control is a research area that covers principles and practices to control costs, schedule, and product planning.
Value-based evaluation	Value-based evaluation is a research area that focuses on techniques to verify and validate that the software solution satisfies its value objectives.
Value-based design and development	Value-based design and development is a research area that focuses on techniques to guarantee that the system's objectives and value considerations are aligned with the business, then inherited by the software design and development practices.

Value-based architecture	Value-based architecture is a research area that comprises the further adjustment of the system objectives with possible architectural solutions.
Value-based requirements engineering	Value-based requirements engineering is a research area that embodies principles and practices to identify stakeholders, identifying the value propositions and reconciling these value propositions into a mutually satisfactory set of objectives for the system.
Value model	Value model represents a business model from an economic perspective, and must determine the economic value exchanged and their inter-venients [104].
Value-based software engineering	Value-Based Software Engineering extends the technical ISO software engineering definition with elements from economics, cognitive science, finance, management science, behavioral sciences, and decision sciences areas [34].
Value constellations	Value constellations is the co-production of values. The business strategy is no longer a matter of positioning a fixed set of activities along with value creating in a value chain. The focus today should be on the value-creating system itself. Where all stakeholders co-produce value [182].

ACRONYMS

ADD	Attribute-Driven Design Method.
ADL	Architectural Description Language.
API	Application Programming Interface.
AQL	Acceleo Query Language.
ATDD	Acceptance-test Driven Development.
BDD	Behavior-Driven Development.
BMeG	Business models for e-government.
BMM	Business Motivation Model.
BMO	Business Model Ontology.
BPMN	Business Process Modeling Notation.
DSL	Domain Specific Language.
DVD	Dynamic Value Description.
EBM	Evidence-based Medicine.
EBSE	Evidence-Based Software Engineering.
EKD	Enterprise Knowledge Development.
EMF	Eclipse Modeling Framework.
EPBE	Eriksson-Penker business extensions.
ETL	Epsilon Transformation Language.
EVL	Epsilon Validation Language.
GBRAM	Goal-Based Requirements Analysis Method.
GMF	Graphical Modeling Framework.
GQM	Goal Question Metrics.
GRL	Goal Requirements Language.
GSN	Goal Structuring Notation.

ACRONYMS

IS	Information Systems.
ISO	International Organization for Standardization.
IT	Information Technology.
ITU	Intention To Use.
KAOS	Keep All Objectives Satisfied.
KAOS4Services	KAOS modeling approach for Service-Oriented Architecture.
M2C	Model to Code.
M2M	Model to Model.
MBE	Model-based Engineering.
MDA	Model-Driven Architecture.
MDD	Model-Driven Development.
MDE	Model-Driven Engineering.
MSc	Master of Science.
NATO	North Atlantic Treaty Organization.
NFRs	Non-functional Requirements.
OMG	Object Management Group.
PEOU	Perceived Ease Of Use.
Ph.D.	Doctor of Philosophy.
PICOC	Population, Intervention, Comparison, Outcome, and Context.
PoC	Proof of concept.
PSS	Product-Service System.
PU	Perceived Usefulness.
QASAR	Quality Attribute-oriented Software Architecture.
RAMA	Reference Architecture Modeling in an Agile software development.
REA	Resource-Event-Agent.

RoI	Return on Investment.
SBMO	Strategic Business Model Ontology.
SECO	Software ECOsystem.
SLR	Systematic Literature Reviews.
SMS	Systematic Mapping Studies.
SOA	Service-Oriented Architecture.
SoaML	SOA Modeling Language.
SOAP	Simple Object Access Protocol.
SS	Systematic Studies.
TAM	Technology Acceptance Model.
TDD	Test-Driven Development.
UCM	Use Case Maps.
UFPE	Universidade Federal de Pernambuco.
UML	Unified Modeling Language.
UNL	Universidade Nova de Lisboa.
UPV	Universitat Politècnica de València.
VBSE	Value-Based Software Engineering.
VLA	Value Level Agreement.
WSDL	Web Services Description Language.
XML	Extensible Markup Language.
XP	eXtreme Programming.

INTRODUCTION

The present research work has its roots in the challenge of deriving a software architecture from an early requirements specification considering the value exchanges of a business. Although much work has been published addressing issues related to software architecture and requirements engineering, no systematic approaches exist to support the aligned construction of an architectural design with the business values that characterize organizations in their marketplace. Such construction process is difficult, requiring skilled and experienced architects. This introductory chapter offers an overview of the context and motivations of this [Doctor of Philosophy \(Ph.D.\)](#) research, defines our problem statement, the involved challenges and corresponding research question, summarizes the major findings, comments on the adopted research methodology, and finalizes highlighting the relationships between the various chapters composing this [Ph.D.](#) thesis document.

1.1 Context and motivation

With a first degree in information systems and 10 years of experience in industry, the major problem I have encountered in my work was related to software architecture, more specifically, how to structure an information system to make it reflect the business needs. Additionally, in the companies I've worked for, the information systems were built based on business processes often not aligned with the companies' business values. These two problems were the motivations for this research work. Therefore, the general goal was to explore systematic and rigorous means to derive a software architectures from early requirements specifications, aligned with the business goals of an organization. The starting point was to survey the existing body of knowledge. To achieve this we followed the best practices of [Evidence-Based Software Engineering \(EBSE\)](#) [149], and performed a tertiary study to catalogue the empirical studies on software architecture aiming at

aggregating the main findings reported in those studies and providing an overview of the consolidated state of the art in software architecture. From the results of this study, we have also surveyed the state of the art on service-oriented architecture (an architectural style commonly used to develop information systems [20, 136, 196]) and identified the absence of a secondary study on software architecture derivation methods, what led us to conduct a systematic mapping study on this particular topic. Both the tertiary study and the systematic mapping study highlighted several challenges that we focus in this Ph.D. research, particularly: (i) the lack of support to build architectural models considering the business values of an organization, and (ii) the strong dependence on the architects' experience and intuition.

1.2 Problem statement

Our problem statement results from the findings discovered in the systematic tertiary and mapping studies. The discussion that follows identifies the two major problems addressed and is structured with a "problem description", a "corroboration" to justify the problem with evidence found in the literature, and a brief "conclusion" to describe what we can do to mitigate or solve the problem.

Problem 1: Lack of support to derive architectural models from business value models

Problem description. Most software architecture derivation methods consider services design as a software engineering problem rather than a business management *and* a software engineering problem [262]. In the area of business management, services must offer value to customers [182, 211]. Most software architecture derivation methods align the software architectures with the *business processes* rather than the *business values* [20, 113].

Corroboration. The [Service-Oriented Architecture \(SOA\)](#) architectural style is widely used to implement [Information Systems \(IS\)](#), thanks to services' low coupling [20, 136]. The critical aspect of developing an efficient SOA solution is both the identification of services from requirements and their mapping into an appropriate service reference architecture [7, 18]. Although some works focus on improving the alignment between business and IS using SOA [20, 32, 129, 154], this is still a challenge mainly due to its multidisciplinary nature [7, 113]. Most of the methods use business process models as the main input for a service architecture design process [20, 113]. Given that services deliver value to their customers, value is, in fact, the reason why companies and people trade with each other. In other words, companies offer something (e.g., money) to get another thing in return (e.g., goods). Thus, deriving a service-oriented architecture that

considers business values is important to identify the software services that better meet the business needs and their respective customers' satisfaction [15, 106].

Conclusion. There is a lack of approaches for deriving architectural models from early requirements that consider the organization's business values, thus offering improved support for the alignment of an IS the software architecture with the business core values.

Problem 2: Software architecture based on experience and intuition

Problem description. The transition from requirements to software architecture is based on the experience and intuition of software architects [24, 96]. Consequently, this transition can hardly be conducted by less experienced professionals.

Corroboration. Deriving a software architecture from an early requirements specification involves many complex decisions [24, 43, 131, 256] that are often based on the experience and skills of the involved architects. This makes the task very difficult for less experienced architects [117]. For example, methods such as [Attribute-Driven Design Method \(ADD\)](#) [266] and [Quality Attribute-oriented Software ARchitecture \(QASAR\)](#) [42] rely heavily on the software architect to find suitable solutions for satisfying quality requirements¹. [21]. Indeed, most architecture derivation methods do not offer sufficient guidelines, relying on a creative process based on, in large extent, the professionals' expertise [24]. Also, it is well accepted that [NFRs](#) are the major drivers of the architectural decision-making process [25, 89, 125]. Although several approaches exist to elicit [NFRs](#) in the early software development stages (e.g., goal-oriented approaches), an interview (empirical) study with 13 architects from 12 companies in Spain² shows that in industry, [NFRs](#) are not handled until the architecture design phase and their elicitation is based on the architects' experience [13]. Another interview with 14 architects found that [NFRs](#) are elicited very late or are never discovered [40]. Although the study participants know the importance of [NFRs](#) these are not considered more important than functional requirements in practice. This may be because customers do not know what [NFRs](#) are, knowing primarily what services they want (or functional requirements) the system to offer, and companies work to deliver a solution to the customers' requests. Thus, we emphasize the importance of eliciting [NFRs](#) in the early stages of software development. A third study with 53 experienced software architects recommends guiding less experienced architects throughout architectural design because of its complexity [117].

Conclusion. The process of building a software architecture from early requirements is strongly based on the architects' experience and intuition, what makes this activity

¹ Quality requirements are a subset of non-functional requirements.

² Although all organizations were based in Spain, some of the projects involved clients from other countries.

difficult for novices. Consequently, there is a need for a systematic, traceable and simple to use approach for the transition between requirements and architecture design.



These the two major problems can be synthesized in the following problem statement:

We lack support to derive software architectures from early requirements specifications aligned with an organization business values, as well as to help novice IT professionals.

1.3 Challenges

The identified problem raises several challenges, in particular, how to develop an **approach to derive architectural models from early business requirements models**, and how to **align the business values of an organization with the resulting software components and structure**. We envision a **systematic and traceable transition** from requirements to software architecture by offering **simple models, and explicit guidelines and mappings** to be followed by **Information Technology (IT)** professionals. The identified challenges are discussed next.

Derive architectural models from early requirements models. **Model-Driven Development (MDD)** automates repetitive and error-prone tasks through an automatic processing **Model** aiming at reducing the effort and accidental complexity involved in software development [155]. **MDD** motivates developers to concentrate on the production of top-level abstraction models to generate software artifacts through automatic, or semi-automatic, transformations of models. These characteristics make it an obvious technological choice to support our software architecture derivation approach. However, developing software through models is complex, requiring a rigorous definition of these models [226] and the semantic correctness of the transformations [153]. The correctness of the transformations is guaranteed by the semantic validation offered by **MDD** tools (e.g., EuGENia [79] and Sirius [80]).

Align the organizations' business values with the resulting software architecture. We argue that business models representing business values can be used during the software requirements elicitation process to facilitate the transition from business to software artifacts. However, there is a wide gap between business models and requirements models, because: (1) requirements engineers find it difficult to extract knowledge from business models to design information systems (business models are complex [162, 212]); and (2) business specialists are not usually aware of commonly used requirements techniques to express system behavior [51]. Therefore, creating requirements models aligned with business models is not straightforward and approaches to realize this alignment automatically

are lacking. A challenge is to avoid losing or misunderstanding valuable information due to communication issues between business and software developers, leading to wrong or needless architectural features [267].

Guide less experienced IT professionals throughout architectural design. To achieve this goal, the detailing of activities at different stages of development (e.g., business, requirements, architecture) is required to provide processes and guidelines facilitating the development process and reuse, as suggested by [117]. A deficiency in the sequence of activities can culminate in the creation of mismatched and misaligned processes, resulting in slowing down the transition activity in detrimental to the intended final result [52]. In other words, it is difficult to balance what really needs to be systematic and how to organize the activities to assist the less experienced IT professionals during the derivation process. In addition, the process of creating guidelines is difficult because it requires deep insight of the problem and its possible solutions [200].

Simple (yet useful) models. The models used should be simple (and still be useful) to be understood and easy to use. IT professionals. For example, the multiple business model origins (e.g., e-Business, business strategy, information systems, business management and economic science) [198], with different researchers defining their concepts in different and overlapping ways, are hard to reconcile to a single definition or understanding platform [8, 192, 198, 271]. As previously mentioned, requirements engineers find it difficult to extract knowledge from business models to design information systems [141]. In fact, a similar problem happens in the transition from requirements models to architectural models [10]. Automating these transitions using MDD is even more challenging because it requires a conceptual mapping between source and target models that belong to different disciplines (often with a different abstraction level as well). We believe that if our proposal is simple, IT professionals will want to use it in the future. This would facilitate its wider adoption, possibly also by the industry.

Backward and forward traceability throughout the architectural derivation process. Due to the market inherent high competition, companies constantly change their needs to provide different and higher quality values to their customers. Such changes at the company's business level should be reflected in the information systems to ensure service quality and customers' satisfaction. Therefore, it is essential to trace important concepts throughout the software development process and among the artifacts created throughout this process. This facilitates the analysis of the impact of business changes on the information system. However, supporting traceability for particular purposes at different stages of the software development is challenging [97].

1.4 Research questions

The problems and respective challenges are structured in Figure 1.1, which offers an overview of the main lines driving our research.

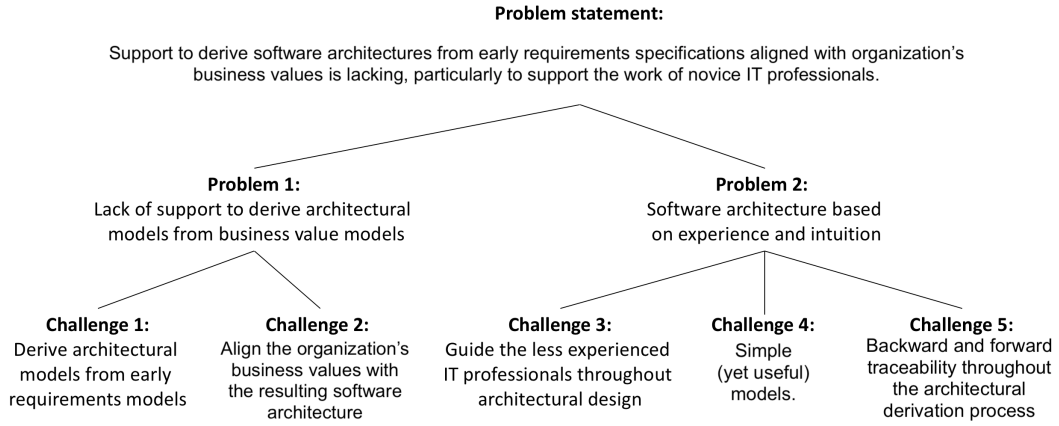


Figure 1.1: Relating problem statement to challenges.

The need to guide less experienced architects in the task of deriving a software architecture from a requirements specification that reflects the organization's business values, leads to the following general research question:

How to derive value-centred architectural models systematically?

Our research must consider both the lack of support to derive architectural models from business value models (problem 1) and the fact that software architectures are built based on the experience and intuition of architects (problem 2). To tackle problem 1, we subdivided the main research question to consider the software architecture derivation from early requirements (challenge 1) aligned with the organizations' business values with the resulting software architecture (challenge 2). The result of this subdivision leads to the following sub-questions (Sub-RQ#):

[Sub-RQ1:] How to **derive** architectural models from early requirements specifications?

[Sub-RQ2:] How to **align** software architecture models with the business values of an organization?

Regarding problem 2, the investigation must consider guiding the less experienced IT professionals throughout the architectural design (challenge 3), offering both, simple (easy to use and useful) models for the software architecture derivation process (challenge 4) and traceability of concepts throughout the derivation process (challenge 5). Thus, as part of the main research question, we should also consider:

[Sub-RQ3:] How to produce **simple models and explicit guidelines** ?

[Sub-RQ4:] How to include **traceability** mechanisms in the architectural derivation process?

1.5 Supporting methodologies, paradigms, and technologies

When we talk about software engineering, the discussion focuses on whether we can consider it science or engineering. This question refers to the double character of software. On the one hand, software engineering considers the process of product creation (software), and from this point of view, it has the specific characteristics of production or engineering [254]. On the other hand, aspects related to time-to-market and competition demand the continuous improvement of process and product quality and delivery. It is in this context that the scientific part of software engineering presents itself [254]. Therefore, specific methodologies, paradigms, and technologies are needed to help establishing an engineering and scientific foundation for software engineering. In this sense, it is worth clarifying which are the supporting methodologies, paradigms and technologies. Our work is based on [EBSE](#) for the principles and guidelines of for literature reviews, [Model-Driven Engineering \(MDE\)](#) for the foundation of the software development process, and experimental methods for the evaluation of our work. These three supporting approaches are described next.

Evidence-Based Software Engineering. [EBSE](#) seeks to provide means by which better evidence from the research can be integrated with practical experience during software development and maintenance [151]. The essence of the evidence-based paradigm is to systematically collect and analyze all available data on a particular phenomenon to obtain a more complete and broader perspective that cannot be grasped by an individual study [151, 185].

The evidence-based paradigm gained strength initially in medicine ([Evidence-based Medicine \(EBM\)](#) [218, 219]), aiming to integrate the best research evidence with clinical experiences and patient assessment. Medical practices have changed dramatically in the last decade through the adoption of evidence-based paradigms, as studies have shown that failure to perform systematic reviews can cost lives [218]. With the increasing importance of software in many areas (e.g., mobile handsets, automated brakes, medical devices, flight control systems), more care must be taken with research methods also in software engineering. Therefore, several researchers (e.g., [78, 151, 185]) suggest that software engineering professionals (and researchers) should consider supporting the use of evidence-based software engineering to create a comprehensive overview of a given research area, to identify the best practices on a given topic, and to improve their decisions about which technologies to adopt.

To offer this support, [EBSE](#) provides a rigorous and reliable research methodology, together with auditing tasks to reduce the researcher bias on the results [45, 149]. Two of the core tools for evidence-based studies are [Systematic Literature Reviews \(SLR\)](#), focusing on identifying the best practices on a given topic based on empirical evidence, and [Systematic Mapping Studies \(SMS\)](#), aiming at creating a comprehensive overview of a given research area [202, 203]. In general, [SMSs](#) can be performed previously to [SLRs](#), to

help identifying research questions for data aggregation. Both [SLRs](#) and [SMSs](#) are considered secondary studies because they aggregate information from primary studies to reveal evidence and build knowledge. In our research, we performed a [SMS](#) to obtain an overview of business modeling (Chapter 2) and another of software architecture derivation methods (Chapter 4). Before performing this second [SMS](#), we conducted a tertiary review to map the most representative secondary studies covering software architecture (Chapter 3). A tertiary review is a systematic study of systematic reviews, hence using the standard methodology for systematic secondary studies. Only after verifying the nonexistence of a secondary study on software architecture derivation methods through our tertiary review, did we execute our second [SMS](#). Some advantages on using these tools are [151]:

- Well-defined methodology: Mitigates biases in literature selection but it does not protect against publication bias in the primary studies
- Information about the effects of some phenomenon across a wide range of settings and empirical methods: If studies give consistent results, systematic reviews provide evidence that the phenomenon is robust and transferable
- With quantitative studies, it is possible to combine (aggregate) data using meta-analytic techniques: Increased likelihood of detecting real effects that individual smaller studies are unable to detect
- Reuse: Researchers can reuse the protocol used in a study to keep the results always up to date for the scientific community

The drawback is that these tools require considerable more effort than traditional ad-hoc literature reviews.

Model-driven Engineering. A common problem in recent years is the growing complexity of the software due to customers' demand for more features and more quality. In response to this demand, software engineering continually offers new methods and tools that, when correctly used, can help in the difficult task of software development. To support our goal, we selected [MDE](#) as it focuses on abstracting the details of a complex problem, concentrating developers on the production of top-level abstract models to generate complex software artifacts automatically or semiautomatically. [MDE](#) is a relatively new software development methodology aiming at raising the level of abstraction to handle the development of complex software systems, has been successfully implemented in many industries, including telecommunication, automotive, aerospace, and business information systems [180]. [MDE](#) automates repetitive and error-prone tasks through an automatic process aiming at reducing the effort and accidental complexity involved in software development [155]. As said, it focuses on abstraction, thus ignoring details of

a complex problem, concentrating developers on the production of top-level abstract models to generate software artifacts automatically.

MDD, **Model-Driven Architecture (MDA)** and **Model-based Engineering (MBE)** are common terms associated with **MDE** [44, 62]. **MDD** is a subset of **MDE**, and it is related to the use of technologies and techniques to operationalize the **MDE** concepts (In a way, this means that **MDE** is a super set of **MDD** because it goes beyond the development activities, encompassing other tasks based on models of a software engineering process, such as model-driven reverse engineering of a legacy system). **MDA** is a subset of **MDE** and **MDD**. It is related to the use of technologies and techniques to operationalize the **MDE** concepts (similar to **MDD**). The difference between **MDD** and **MDA** is that the later only uses technologies and techniques proposed by the **Object Management Group (OMG)**. Finally, **MBE** is used to refer to a light version of the **MDE**. It is a process in which models play an important role, although they are not necessarily the key development artifacts. An example would be a development process in which in the design phase the various models of the system are created, but then the models are delivered to the programmers so that the code is written manually (without automatic code generation). In this process, models continue to play an important role, but they are not the main artifacts of the development process [44].

The **MDE** core concepts are model, metamodel, and model transformations [44, 62]. They are presented below. A **Model** is an abstract representation of a system (or a theory or reality) that allows one to make inferences and predictions about the system [163]. **MDE** uses *models* as first class entities, with the advantage of increasing productivity, augmenting interoperability, and facilitating communication [16, 222]. However, developing software through models requires a rigorous definition of these models [226]. A **Meta-model** is a template for defining templates [177]. A model is an instance of a metamodel, and a metamodel comprises all models that can be expressed through it [177]. In other words, a metamodel is who strictly defines the models. Finally, **model transformations** are made taking into account the source models and target models. In order to make a transformation, it is necessary to map the concepts of the metamodels of the respective models involved in the transformation [177]. That is, transforming a model A into a model B requires a mapping between the concepts of the metamodel of model A and the concepts of the metamodel of model B. In this way, the transformation of model A generates a new model B. Figure 1.2 illustrates the transformation scheme. Model transformations automatically refine, refactor or re-engineer source models [153]. Hence, models are incrementally refined through transformations — known as **Model to Code (M2C)** and **Model to Model (M2M)** —, starting from a problem model (the source model) until the production of a solution model (the target model).

One way to implement **MDE** is through the construction of **Domain Specific Language (DSL)**s. A **DSL** is a language designed to be useful for a specific set of tasks and a particular domain [109], realize a particular point of view of a problem, and create a rigorous model editor. What differs fundamentally from a **DSL** of a general purpose language (e.g., Java)

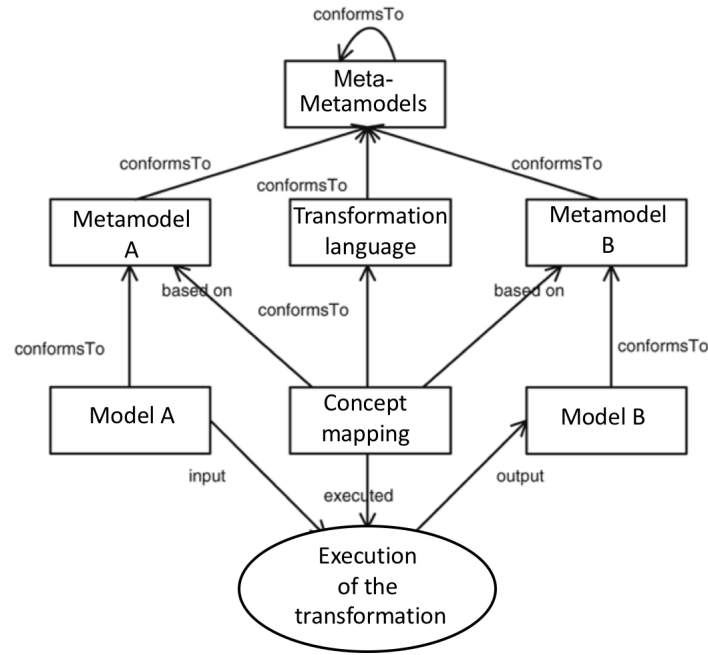


Figure 1.2: Transformation scheme [137].

is that the former is developed from the domain of the problem and not from the domain of the solution. Ideally, each element of the DSL structure is directly related to a concept present in the problem domain, and each restriction of that domain refers to one or more constraints in the language. Adherence between DSL and the problem domain provides many benefits, such as increased productivity, abstraction level, and system quality [145]. This adherence, however, determines its uselessness in other domains [145].

A DSL, like any language, must have syntax and semantics. Syntax defines its structure and semantics defines its meaning. The syntax of a language is divided into two: abstract and concrete. The *abstract syntax* is usually specified in a metamodel, and it defines the language constructs, their properties, and their relationships. The *concrete syntax* defines a notation for the concrete representation of the various concepts of the abstract syntax through drawings, tables, matrices or simple text. The chosen form of representation must be in sync with the concepts of the domain. This is called the principle of representational fidelity [261]. The *principle of representational fidelity* predicts that only one form of representation for each concept of the domain simplifies the definition of notation and also ensures that all concepts can be represented unambiguously in the language [145].

The semantics of a language is the definition of the meaning of the language constructs. Because a language is specific to a particular domain, most of its elements carry meaning from the domain. For example, in the development of mobile software, the Camera and Display concepts have well-defined semantics. Since the semantics of a general-purpose language is not related to a business domain, it is up to the developers to describe the

domain to the semantics of the language.

Experimental evaluation methods. There are four relevant methods for conducting experiments in the area of software engineering [265]: scientific, engineering, experimental, and analytical.

The scientific method observes the world, suggests the model or theory of behavior, measures, and analyzes to verify the hypotheses of the model or theory. This is an inductive paradigm. This method tries to extract from the world some model that can explain a phenomenon and to evaluate if the model is representative of the phenomenon under observation. This was the method we used to build our models (e.g., business model).

The engineering method looks at existing solutions, suggests the most appropriate one, develops, measures and analyzes, and repeats until no further improvement is possible. It is an evolutionary improvement-oriented approach that assumes the existence of some model of the software process or product and modifies this model to improve the objects of the study. We used this method when selecting existing approaches (e.g., goal-oriented requirements engineering) and developing proof-of-concept tools using DSL editors (Chapter 5).

The experimental method creates the model (solution), develops the qualitative and/or quantitative method, applies an experiment, measures, analyzes, and evaluates the solution, and repeats the whole process. This method is an approach oriented towards revolutionary improvement. The process begins with the survey of a new solution, not necessarily based on an existing solution, and studies the effect of the process or product suggested by the new solution. We used this method to perform controlled and quasi-experiments to validate our research (Chapter 7).

Finally, analytical (or mathematical) method suggests a formal theory, develops that theory, derives the results, and if possible, compares it with empirical observations. This method is a deductive approach that does not require experimental design in the statistical sense but provides an analytical basis for developing solutions. From the four methods explained, the analytical is the only one we did not use.

1.6 Major results

Our major result is a **framework** to derive a software reference architecture model from business value models. This framework includes:

- **A set of guidelines, set of model transformation schemes, methods, and processes** to derive a software reference architecture model from business value models. In particular, we created the methods [Dynamic Value Description \(DVD\)](#), [Reference Architecture Modeling in an Agile software development \(RAMA\)](#) and [KAOS modeling approach for Service-Oriented Architecture \(KAOS4Services\)](#) to support an

integrated step-by-step and systematic development, using each method independently or in tandem:

- *DVD* method for business value modeling. *DVD* is **easy to use**, **useful** for business modelers, and has sufficient **efficacy** to be used during the derivation of a software architecture [240, 241, 243].
 - *Reference Architecture Modeling for Agile development*. *RAMA* software development is a value-centric method to address the architecture-agility combination.
 - *KAOS4Services* for traditional software development. *KAOS4Services* method is a systematic approach to modeling *SOA* applications using goal-oriented models.
- **Traceability support** of the relevant concepts (e.g., value, *NFRs*) from the business value modeling phase down to the software reference architecture model. The fundamental conceptual and technological infrastructure are provided by the *MDD* techniques.
 - **A set of evidence-based studies** (empirical studies) to validate various parts of our proposal, contributing to the current body of knowledge.
 - A quasi-experiment to evaluate the method of *DVD* in relation to its ease of use and utility.
 - A family of three controlled experiments and a meta-analysis to compare the *DVD* method against the *e3value* method (a widely used business value modeling method) with respect to its efficiency, perceived efficacy (ease of use and utility) and intention to use the methods in the future by the participants.
 - A quasi-experiment to evaluate the *RAMA* and *KAOS4Services* methods regarding their easy to use, utility and intention of participants to use the methods in the future. In addition, we did a comparative analysis between the two methods.
 - **Prototyping tools** to business, requirements, and *SOA* modeling tasks through *DSLs*. We automated some error-prone tasks, such as the transition between business modeling to requirements modeling through model transformations (for example goal-oriented *Keep All Objectives Satisfied* (*KAOS*) models are generated from the *DVD* language).
 - Six scientific papers published in international conferences (C#), and two journal articles (J#). Table 1.1 summarizes the goal of each paper and offers its classification

according to CORE-ERA³ and SJR⁴, as well as the Brazilian Qualis⁵ for conferences and journals).

Table 1.1: List of publications.

#	Title	Venue	Goal of the paper	CORE-ERA or SJR / Qualis
C1	Comparing Value-Driven Methods: an experiment design [238]	2nd International Workshop on Human Factors in Modeling (Hu-FaMo'16@MODELS'16)	The design of an experiment to compare two business value methods (e3value and DVD)	- / B2
C2	An approach to align business and IT perspectives during the SOA services identification [240]	17th International Conference on Computational Science and Its Applications (ICCSA 2017)	A business value modeling approach that uses model-driven techniques to generate the input required by current software services identification methods, thus aligning business and software perspectives	C / B1
C3*	Aligning business models with requirements models [239]	European Mediterranean & Middle Eastern Conference on Information Systems (EMCIS2017)	A systematic approach to automatically generate goal-oriented models from value models	B / -
C4	Evaluating the efficacy of value-driven methods: a controlled experiment [241]	26th International Conference on Information Systems Development (ISD2017)	A controlled experiment comparing the DVD method with the well-known e3value method, with respect to their effectiveness, efficiency, perceived ease of use, perceived usefulness and intention to use	A / B3
C5	Towards an Agile Reference Architecture Method for Information Systems [244]	Hawaii International Conference on System Sciences 2018 (HICSS2018)	The RAMA (Reference Architecture Modeling in an Agile software development), a value-centric method to address the architecture-agility combination	A / A1
C6	Deriving Services using KAOS Models [237]	33rd ACM/SIGAPP Symposium On Applied Computing (SAC2018)	KAOS4Services, a systematic approach to derive services from goal-models expressed using the KAOS language	B / A1

Continues on next page

³<http://portal.core.edu.au/conf-ranks/>

⁴<https://www.scimagojr.com/journalrank.php>

⁵<https://sucupira.capes.gov.br>

Table 1.1 – Continuation from previous page

#	Title	Venue	Goal of the paper	CORE-ERA or SJR / Qualis
J1	Comparing Business Value Modeling Methods: A Family of Experiments [243]	Information and Software Technology Journal (IST), 2018	A family of three controlled experiments to compare two different value-driven methods, offering empirical evidence regarding the methods' efficacy when modeling business value and their likelihood of acceptance in practice.	Q2 / A2
J2	Deriving Architectural Models from Requirements Specifications: a Systematic Mapping Study [245]	Information and Software Technology Journal (IST), 2019	A comprehensive overview of the existing methods to derive architectural models from requirements specifications, offering a research roadmap based on the identified limitations and open issues.	Q2 / A2

* means Best Paper Award of the conference.

1.7 Research methodology

Our methodological research approach is based on the Technology Research method [232], which encourages the creation or improvement of artifacts to address a specific need. The main steps of this methodology are:

- *Problem analysis* addressing “*what is the potential need*” for a new technology. In this step researchers identify a need for new or better artifacts.
- *Innovation* addressing “*how to make an artifact that satisfies the need*” identified during the problem analysis step. In this step researchers produce new or better artifacts.
- *Validation* addressing “*how to show that the artifact satisfies the need*”. In this step researchers check if the artifacts created during the innovation step satisfy their requirements. When new artifacts are obtained, the researcher has a basis for new questions, leading to new investigations. Therefore, technology research is an iterative process.

Next, we discuss how the technology research process was followed in this research work.

The problem analysis step was accomplished performing systematic studies of the literature to obtain a full coverage of the three topics of interest: business value modeling,

software architecture methods, and software architecture derivation methods. We started with a systematic mapping study to build the state of the art on business modeling. This study showed the complexity of modeling business values to be used in the construction of information systems' architectures. Then, we performed a tertiary study on software architecture to create a catalog of consolidated software architecture methods, techniques and tools subject of the selected systematic secondary studies. From the catalog, we realized the nonexistence of secondary studies on derivation methods of software architecture from requirements specifications, what lead us to carry out the final systematic mapping. From the set of findings extracted from this systematic mapping and the tertiary study, we identified several problems and challenges with the transition between requirements and software architecture. The results of this phase will be discussed in some detail in Chapter 2, Chapter 3, and Chapter 4.

The *innovation* step should explain how to satisfy the needs identified during the *problem analysis*, and to develop the required efforts to create a solution. We created a framework for the derivation of a software reference architectural model considering the organization's business values. This framework is composed of four large modules (business modeling, agile requirements modeling, goal-oriented requirements modeling, and software architecture) organized in two layers (business and software layers). The main purpose of the business modeling module is to build value models for an organization. It generates [SOA Modeling Language \(SoaML\)](#) models to represent the business value exchanges with a software engineering perspective. The requirements modeling modules aim at decomposing the business models for identification of software [Service](#). The software architecture module aims at generating architectural models.

The *validation* step was performed in several different ways, that ranged from applying the techniques to examples (e.g., e-commerce), to developing case studies (e.g., online auction system), to building prototypes as proof of concepts through [DSLs](#), to performing a quasi-experiment to check if our business model is perceived as easy to use and useful as well as a family of controlled experiments to compare the [DVD](#) method with the e3value [104] (a well-known business value modeling method). The results of this work have also been peer-reviewed and published in relevant fora, hence also validated by the feedback obtained from the reviewers.

1.8 Structure of this document

This document is composed of eight chapters, technically aggregated in five major parts, as depicted in Figure 1.3, where: the first part is this introduction; the second part is composed of chapters 2-4 and reports on the state of the art about business modeling methods and software architecture derivation methods; the third part is composed of Chapter 5 and addresses the value-based framework for software architecture; the fourth

part is composed of chapters 6-7 and tackles the validation of this [Ph.D.](#) research; finally, the fifth part is the concluding chapter. Each chapter is summarized next.

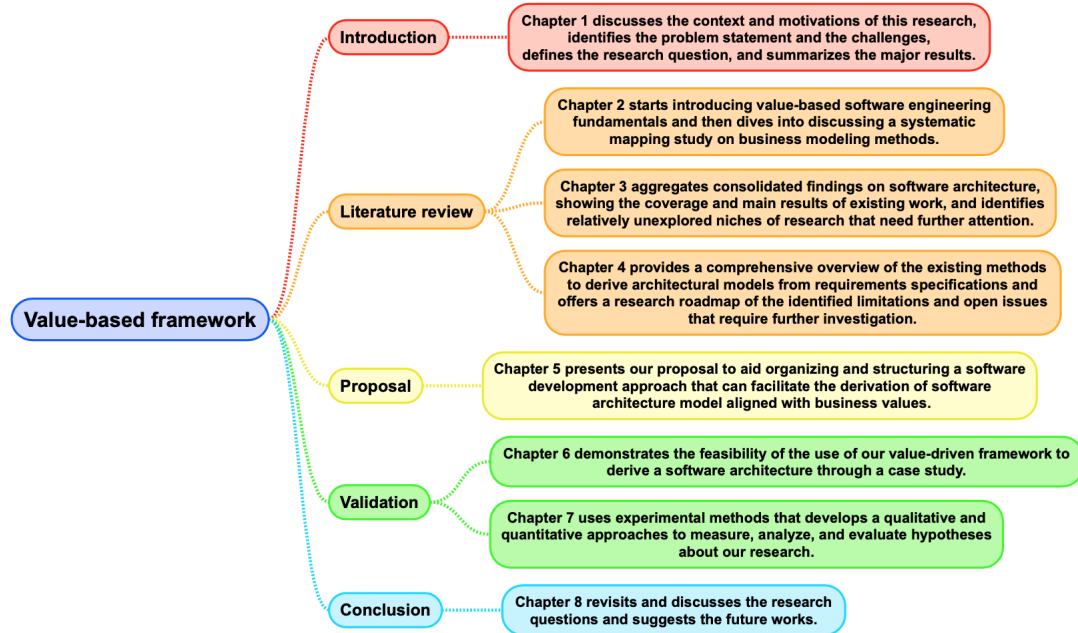


Figure 1.3: Document structure.

Chapter 1, *Introduction*, discusses the context and motivations of this [Ph.D.](#) research, identifies the problem statement and the associated challenges, defines the research question, summarizes the major results, and overviews the adopted research methodology.

Chapter 2, *Business modeling*, starts by introducing value-based software engineering fundamentals and then dives into discussing a systematic mapping study on business modeling methods aiming at identifying which existing methods are used to create models representing business value exchanges. We argue that the idea of value-based software engineering is essential for an appropriate alignment between business and computed information systems. Our study emphasizes in the value-based requirements engineering and value-based architecture. Complementing the idea of value-based software engineering, we explain the basic concepts needed to represent business value through a model.

Chapter 3, *Software architecture*, aggregates consolidated findings on software architecture, showing the coverage and main results of existing work, and identifies relatively unexplored niches of research that need further attention. To achieve this goal, we performed a tertiary study on software architecture, aiming at gaining a better understanding of its most notable findings and existing challenges, as reported in the literature. We identified, for example, the nonexistence of secondary studies looking for software architecture derivation methods from requirements specifications. This result lead us perform a systematic mapping on this topic (see Chapter 4).

Chapter 4, *Deriving architectural model from requirements specifications: Software architecture derivation methods*, provides a comprehensive overview of the existing methods

to derive architectural models from requirements specifications and offers a research roadmap of the identified limitations and open issues that require further investigation. The systematic mapping study follows the good practices from the [EBSE](#) field. The major findings indicate that current architectural derivation methods rely heavily on the architects' tacit knowledge (experience and intuition), do not offer sufficient support for inexperienced architects, and lack explicit evaluation mechanisms. These and other findings are synthesized in the research roadmap, which we hope will benefit researchers and practitioners.

Chapter 5, *A value-driven framework for software architecture*, presents our proposal for the derivation of software architecture models aligned with business values, offering explicit methods, model languages, guidelines, mappings and tools to help the architects. This framework is composed by the methods [DVD](#), [RAMA](#), and [KAOS4Services](#). The [DVD](#) method captures the key concepts of business values in models easy to build and simple to understand by business and [IT](#) professionals. [RAMA](#) is used to derive a software architecture model from a [DVD](#) model through the agile software development cycle. Finally, [KAOS4Services](#) is also used to derive a software architecture model from a [DVD](#) model; it follows a “more traditional” software development using a goal-oriented approach. Each of these methods offer a model language, each implemented as a [DSL](#), and a step-by-step process to help applying the methods.

Chapter 6, *Case Study*, uses an industrial online auction system, that is part of a Brazilian gas station chain fidelity program, to demonstrate the use of our value-driven framework to derive a software reference architecture model. It starts by applying the [DVD](#) method to build a value model, from where the subsequent application of the [RAMA](#) or the [KAOS4Services](#) methods results in the derivation of a reference software architecture model.

Chapter 7, *Evaluation through experiments*, uses an experimental method that develops a qualitative and/or quantitative approach to measure, analyze, and evaluate hypotheses. This method is considered the most appropriate approach to experimentation in the area of Software Engineering [254]. This Chapter describes a quasi-experiment to evaluate the perceived efficacy (ease to use and usefulness) of the business layer of the framework ([DVD](#) method), a family of the three controlled experiments and a meta-analysis to compare the methods [DVD](#) and [e3value](#)⁶ with respect to their actual efficacy (effectiveness and efficiency), perceived efficacy (perceived ease of use and perceived usefulness), and intention to use. Regarding the software layer of the framework, this Chapter presents a quasi-experiment used to evaluate the [RAMA](#) and [KAOS4Services](#) methods (individually and comparatively) for the purpose of verifying the perceived efficacy of the methods with respect to their perceived ease of use and usefulness, from the point of view of software engineers, in the context of professionals with a background in computer science (with full or on-going [IT](#) courses) and who have participated in at least one information

⁶[e3value](#) is a well-known business modeling approach.

systems software development project in industry.

Finally, Chapter 8, *Conclusions*, finishes the document by revisiting and discussing the research questions in the light of the results accumulated in the earlier chapters, and, for each research question, also highlighting the contributions of the work. We also offer an overview of the current status of the work, list the articles and papers published, and end with a discussion of our intentions and visions for future work.

BUSINESS MODELING

There is a widespread agreement about the importance of business models for a company to express its [Value](#) exchanges, be them economic, social, or other [106, 212, 263]. Therefore, it seems reasonable to expect an alignment between the company’s information systems and the business values expressing its economic perspective. This alignment can be achieved by using the business economic values expressed in a business model to guide the software development process. This chapter presents background information on business modeling by offering an overview of the main concepts and representations, discussing the process and results of performing a [SMS](#), and presenting two different approaches to business modeling selected based on the results of our systematic study. The major findings indicate that current business modeling approaches are difficult to be used by software engineers and do not provide the necessary rigor for software development.

2.1 Overview on business modeling

As the general goal of our work is to investigate how to derive architectural models aligned with business values, this section presents an overview of business modeling and shows its relation with the [IT](#) research area.

2.1.1 Value-based software engineering

The [International Organization for Standardization \(ISO\)](#) defines software engineering as “*the systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, testing, and documentation of software to optimize its production, support, and quality*” [126]. While this definition might serve the purposes of some software projects there will be others with an higher impact on the business value of a company, for which this definition must be extended to consider business value.

For example, the ISO definition excludes the fields of economics, management science, cognitive sciences, and humanities from the body of knowledge required to create successful software systems (mainly to build information systems) [36]. The ISO definition also delimits the software development by technical activities (e.g., design, implementation, and testing) and it does not explicitly recognize the ultimate goal of software development: “*ensuring that software systems continue to meet and adapt to evolving human and organizational needs to create value*” [34, 36].

In contrast, **Value-based software engineering** considers software development as a purposeful activity carried out by people for people, without ignoring the body of knowledge of those fields. In addition to the technical activities described in ISO, VBSE also regards management-oriented activities as part of the software engineering lifecycle (e.g., business case development, project evaluation, project planning, process selection, project management, risk management, process measurement, and monitoring) [36, 139, 144]. According to Boehm [36], VBSE definition includes “*the explicit concern with value in the application of science and mathematics by which the properties of computer software are made useful to people*”, and its major elements are [36]:

1. **Value-based requirements engineering**, embodying principles and practices to identify stakeholders, identifying the value propositions and reconciling these value propositions into a mutually satisfactory set of objectives for the system (e.g., [111, 140]);
2. **Value-based architecture**, comprising the further adjustment of the system objectives with possible architectural solutions (e.g., [90]);
3. **Value-based design and development**, involving techniques to guarantee that the system’s objectives and value considerations are aligned with the business, then inherited by the software design and development practices (e.g., [258]);
4. **Value-based evaluation**, including techniques to verify and validate that the software solution satisfies its value objectives (e.g., [100]);
5. **Value-based planning and control**, covering principles and practices to control costs, schedule, and product planning (e.g., [35]);
6. **Value-based risk management**, combining principles and practices to identify, analyze, prioritize and mitigate risk (e.g., [67, 100]);
7. **Value-based quality management**, involving prioritization of desired quality factors concerning the stakeholders’ value propositions (e.g., [49]);
8. **Value-based people management**, including managing the expectations, as well as the project’s accommodation of all stakeholders’ value propositions (e.g., [60]);

9. *Value-based software engineering theory*, combining the traditional computer science theories with value-based theories (e.g., utility theory, decision theory, dependency theory, and control theory) to provide processes and framework for guiding VBSE activities (e.g., [37]).

Our research contributes to *value-based requirements engineering* by providing a business ontology supported by the DVD method and a business Value model to facilitate business modeling focused on value exchanges between actors (details in Chapter 5). We also contribute to *value-based architecture* with the agile RAMA method and the goal-oriented KAOS4Services method (details in Chapter 5).

2.1.2 Value-driven modeling

Business modeling covers several different disciplines such as information systems, business management, information technology, economics, business strategy and e-commerce [212]. Despite the growing importance of business modeling to the success of organizations, its multi-disciplinary origins results in a lack of cohesion and understanding of its definition, fundamental concepts, components and taxonomy of business models [8, 271]. Business modeling can be regarded as a conceptual tool to express the main business logic of an organization, describing the set of values that the organization offers to its clients and the network of its partners with which it exchanges values in a way that is sustainable and cost-effective [192]. It is worth noting the difference between business modeling and process modeling [106]. The goal of a process model (e.g., Business Process Modeling Notation (BPMN) [263]) is to clarify *how* processes should be carried out, and by *whom* [106]. In contrast, a business model main goal is to identify *who* is offering *what* to *whom* and expects *what* in return [106]. The central notion in a business model is the concept of *value*, to explain the creation, addition, and the exchange of value between stakeholders [106]. Value is the reason why companies and people trade with each other, exchanging goods among them. Therefore, a *value model* represents a business model from an economic perspective, and must determine the economic *value exchanged* and their intervenients [104]. A Business model can be used as an efficient way of understanding, evaluating, managing and conveying the core concepts of a business [107, 198]. It can be considered the basis to design an information system that supports the needs of a company [271].

2.1.2.1 Basic concepts of a business value model

Business organizations offer services or products (or both) of economic value to its clients, or actors (e.g., in e-commerce), and receive something of value in return. Thus, the basic concepts in business modeling are actors, resources, and the transfer, or exchange, of resources between actors [15, 104]. Therefore, value proposition expressions are important requirements that underpin the business idea. Consequently, the requirement

expressions seen from an economic value perspective respond to the following questions [104]:

1. Who are the business actors involved? It is important to distinguish the actors, so that each one can be satisfied in the best way possible.
2. What objects of economic value are created, exchanged and consumed by these actors? Each actor must know what valuable products it need to produce and consume.
3. What do actors expect in return for an item of value delivered? Actors aim at profiting, usually by getting a valuable object (e.g., money) in exchange for an object they deliver.
4. What phenomena cause exchanges of objects between actors? For example, an actor buys a specific product in a given store because of its fast delivery service. So, “fast delivery” is the phenomena that triggers the value exchange between the actor and the store.
5. What are the activities needed to specify value creation? The activities serve to specify the operationalization of creating or adding value.

2.1.2.2 Visual business value approaches

Value Chain [207] and Value Maps [250] are two visual approaches often used to specify business value models [104].

The value chain approach. A typical and more common use of the value chain notation is illustrated in Figure 2.1 [38]. It shows a sequence of the process of adding value to a network-based value chain (from the manufacturer to the consumer). A value chain approach lacks expressive power for a software requirements specification. This approach does not show who is exchanging value objects with whom (it shows only the sequence of value adding processes, which is not the same thing), does not present the value objects nor does it recognize the concept of economic reciprocity. In summary, although it is a good approach for a general overview of the values of a business, it does not offer enough information to serve well software development.

Value maps approach. A value map shows actors and exchanges of tangible and intangible value objects (e.g., goods, services, revenues, knowledge and intangible benefits) [250]. Figure 2.2 shows an illustrative example for the Cisco system using the Value map approach. Value maps are good for quickly drawing (e.g., on a whiteboard during brainstorm sessions) but cannot express [104]:

- Who is offering what to whom and expects what in return (economic reciprocity);

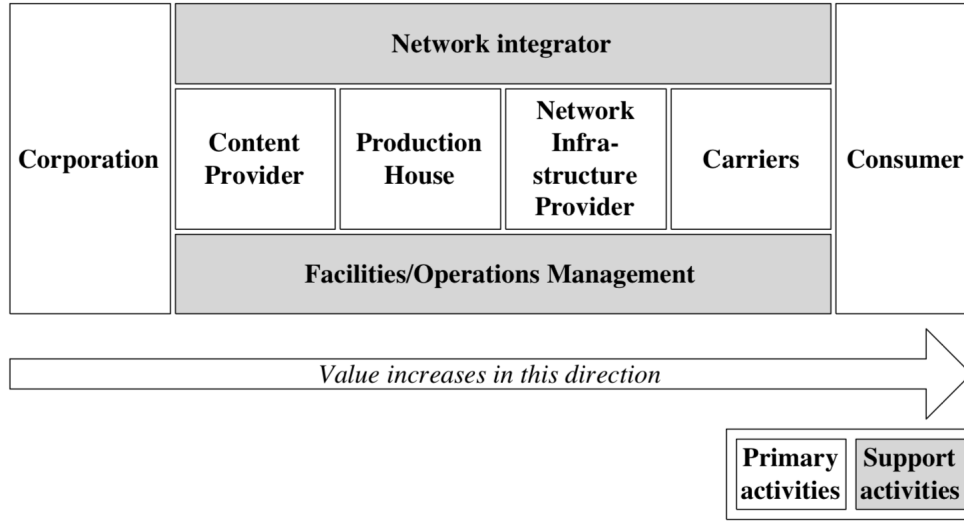


Figure 2.1: A company that offers its product to a consumer using a content provider, a producer (or production house), a network infrastructure provider, and some carriers (e.g., a telecom operator). It uses a network integrator (layer above) and facilitates the management of operations as support (layer below) — details in [38].

- Who is performing each value activity (only actors are recognized);
- Bundles of value objects;
- Partnerships of actors.

Moreover, value maps do not distinguish stakeholders' perspectives very well because there is no explicit focus on valuable objects [104]. In sum, it uses a combination of simple, unstructured, informal textual elements and graphical notations. This combination of elements impacts negatively in the understanding of the business model, often perpetuating the existing misunderstandings among stakeholders (e.g., the business specialist and the IT professional) who must create the initial artifacts for the later implementation of the information system [108]. For example, when we look at Figure 2.2, it is difficult to say if all the arrows between two actors are related to one or more exchanges of values.

2.2 A mapping study on business models

Our goal is to identify the existing methods used to create models representing business value exchanges. To achieve this goal, we performed the three phases of the *Systematic Studies (SS)* process [149]: planning, conducting, and reporting. The *planning* phase defines and evaluates the research questions and research protocol of the study. The *conducting* phase searches relevant primary studies, and extracts and synthesizes the data found according to the protocol. Finally, the *reporting* is concerned with the dissemination of the results. We detail each phase in the following sections.

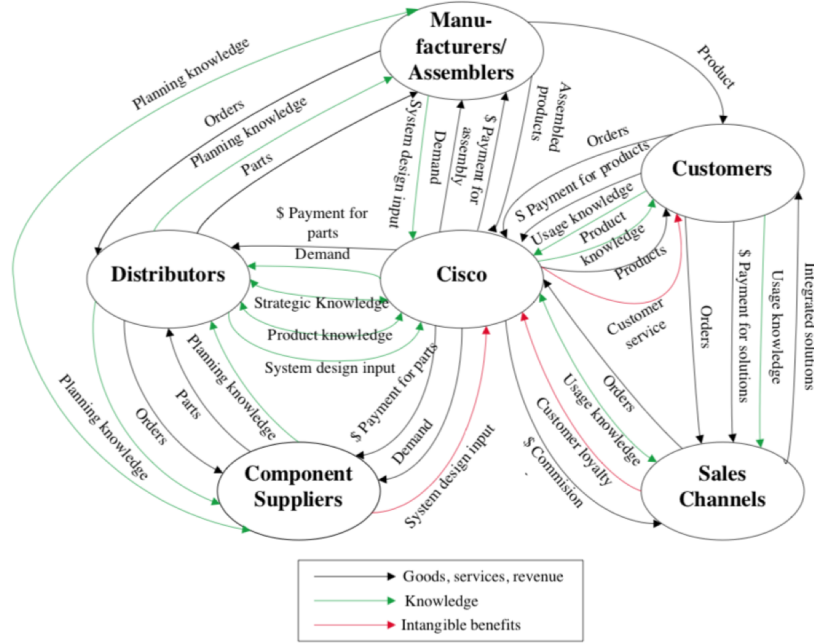


Figure 2.2: Cisco outsources the manufacturing of its products through Component Suppliers (who produce the electronic components), Manufacturers/Assemblers (who produce the Cisco’s products), and distributors (who transports the electronic components from the Component Suppliers to the Manufacturers/Assemblers). With the products in hand, Cisco sells them using Sales Channels (electronics stores) or directly to the Customers (details in [250]).

2.2.1 Planning: research protocol

The various elements of the planning phase are represented in Figure 2.3 and discussed next, starting with the activity “*search and analyze related works*”.

Search and analyze related works. The goal of this activity is to look for “competing” reviews and analyze all existing evidence published by these reviews on the topic of interest. The outcome of this activity was a literature review published in 2012 identifying twelve methods for modeling value concepts [162]. Given the existence of this study, the scope of our systematic review was focused to search for primary studies published in the period from 2011 to 2017, updating the list of those twelve methods.

Define research questions. We did not use any particular approach to create search queries (e.g., PICOC [205]), as the goal of our study is that of the study in [162]. Thus, the main research question is:

What methods are there to specify business models and what are their characteristics?

This main research question was decomposed into the following five sub-research questions to be answered with data extracted from each method found selected:

RQ1: What are the research areas covered by this particular method?

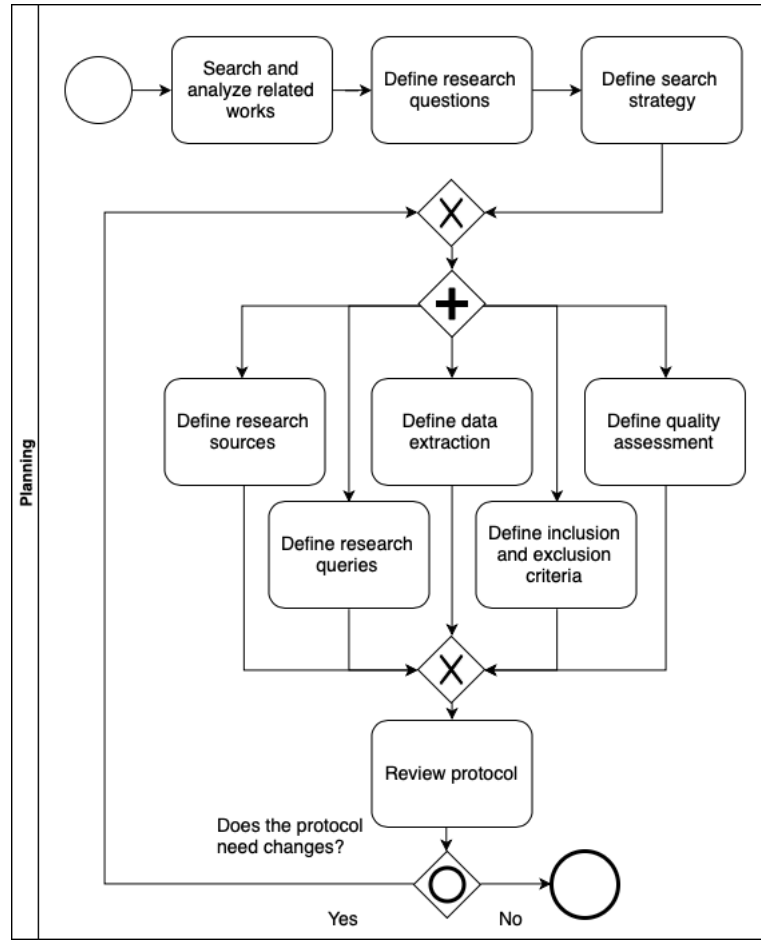


Figure 2.3: Planning phase.

RQ2: What are its main concepts?

RQ3: What is its main goal?

RQ4: Is it supported by a tool?

RQ5: Does it use a graphical or textual notation?

Regarding the RQ1, business modeling covers several different research areas such as information systems, business management, information technology, economics, business strategy and e-commerce [212]. Thus, the goal of this research question is to identify the origin of the selected methods. Regarding RQ2, as the models cover several different research areas, they can represent different concepts to address different purposes. Hence, the goal of this research question is to identify the concepts described in the models created by the methods. Regarding RQ3, the goal is to identify the main reason that led to the creation of the method. Regarding RQ4, a tool is essential for the method to be used in practice. Finally, regarding RQ5, a graphical notation can facilitate communication and understandability of the concepts (e.g., road signs).

Define search strategy. The search strategy used was *automatic*, using search terms to find primary studies in digital libraries, as described next.

Define research sources. We executed the search string in digital libraries search engines to obtain studies that answer the research questions. The digital libraries chosen, chosen for being the most used by the scientific community of the area, were: ACM digital library [5], IEEEExplore [124], and Science Direct [75].

Define data extraction. The data relevant to our research was extracted from the primary studies to a spreadsheet workbook previously structured as a form. We extracted data showing characteristics of the included primary studies (i.e., research source, title, authors, year of publication, and venue) and information useful to answer our research questions (i.e., method's name, source discipline, main concepts, main goal, support tool, and graphical notation).

Define quality assessment. It is common to use some quality criteria in the selection of primary studies during systematic reviews, although the evidence-based software engineering community still does not clearly define what is a study with quality [151]. However, as we are replicating the study by Kundish and John [162] to update their results, and their study does not describe the use of any quality criteria, we should follow the same process.

Define research queries. Since we want to select primary studies to represent **business models** and the central notion in a business model is the concept of **value** (as described in Section 2.1.2), we selected the following set of keywords to create the string to search for studies in digital libraries: *business model*, *business modeling*, *business modelling*, *value chain*, *value delivery*, *value model*, *value modeling*, *value modelling*, and *value network*. The search string uses logical operators *AND* and *OR* to connect the various terms, as follows:

“((‘*value model*’ OR ‘*value network*’ OR ‘*value delivery*’ OR ‘*value modeling*’ OR ‘*value modelling*’ OR ‘*value chain*’) AND (‘*business model*’ OR ‘*business modeling*’ OR ‘*business modelling*’))”

Define inclusion and exclusion criteria. Inclusion and exclusion criteria were defined to help selecting the studies for analysis. We have included all selected studies that were listed in [162] (I1). In addition, we included all papers from journals, conferences and workshops returned by the digital libraries (I2). On the other hand, we excluded informal literature (slide shows, conference reviews, informal reports), secondary and tertiary studies (reviews, surveys) and studies from conferences, workshops and journals without peer-review (E1), duplicated studies or studies with the same content (E2), studies that did not answer the research question (E3), and studies that were not written in English

(E4). In cases of studies complementing previous work, only the more recent one were selected, excluding the older study as duplicate (E2).

2.2.2 Conducting: search results

The execution of the search string in the three digital libraries retrieved a total of 1438 candidate primary studies, which were collected and imported into a spreadsheet workbook. The resulting select the primary studies by applying the inclusion and exclusion criteria, a step that decreased the number of papers to 18 relevant studies. Table 2.1 summarizes this process.

Table 2.1: Application of the Filtering Criteria in business modeling research.

Criteria	From [162]	IEEE	ACM	Science Direct
I1	+12	+0	+0	+0
I2	+0	+54	+21	+1363
E1	-0	-3	-0	-42
E2	-0	-7	-4	-111
E3	-0	-44	-15	-1199
E4	-0	-0	-0	-7
Total	12	0	2	4

I1 - Included papers from [162].

I2 - Included relevant studies cited by authors.

E1 - Excluded informal literature and secondary and tertiary studies.

E2 - Excluded duplicated studies or studies with the same content.

E3 - Excluded studies that did not answer the RQs or that were not available for download.

E4 - Excluded studies that were not written in English.

The 18 selected primary studies were read fully and the relevant data was extracted and added to a spreadsheet workbook previously structured as a form. The list of all selected papers can be found in Table 2.2, where the first twelve were selected from [162].

Table 2.2: Selected studies for the mapping study on business modeling.

Id	Paper
S1	Porter, Michael E. "What is strategy." Harvard Business Review, 1996, pp. 61-78.
S2	Peinel, G., Jarke, M., and Rose, T., Business models for eGovernment services, Electronic Government, an International Journal, 2010, pp. 380-401.

Continues on next page

Table 2.2 – Continuation from previous page

Id	Paper
S3	Osterwalder, A., The business model ontology: A proposition in a design science approach, University of Lausanne, 2004.
S4	Casadesus-Masanell, R. and Ricart, J. E., From strategy to business models and onto tactics, Long Range Planning, 2010, pp. 195-215.
S5	Gordijn, J. and Akkermans, H. M., Value-based requirements engineering: Exploring innovative e-commerce ideas, Requirements Engineering, 2003, pp.114-134.
S6	Weill, P. and Vitale, M. R., Place to space: Migrating to ebusiness models, HBS Press, 2001.
S7	Eriksson, H. E. and Penker, M., Business modeling with UML, Wiley, 2000.
S8	McCarthy, W. E., The REA accounting model: A generalized framework for accounting systems in a shared data environment, The Accounting Review, 1982, pp. 554-578.
S9	Samavi, R., Yu, E., and Topaloglou, T., Strategic reasoning about business models: A conceptual modeling approach, Information Systems and E-Business Management, 2009, pp. 171-198.
S10	Tapscott, D., Lowy, A., and Ticoll, D., Digital capital: Harnessing the power of business webs, HBS Press, 2000.
S11	Parolini, C., The value net: A tool for competitive strategy, Wiley, 1999.
S12	Pynnonen, M., Hallikas, J., and Savolainen, P., Mapping business: Value stream-based analysis of business models and resources in information and communications technology service business, International Journal of Business and Systems Research, 2008, pp. 305-323.
S13	Handoyo, Eko, Slinger Jansen, and Sjaak Brinkkemper. "Software ecosystem modeling: the value chains."Proceedings of the Fifth International Conference on Management of Emergent Digital Ecosystems. ACM, 2013.
S14	Agbabiaka, Olusegun, and Gbenga Adebusuyi. "Delivering eGovernment services through the eTrade distribution network."Proceedings of the 5th International Conference on Theory and Practice of Electronic Governance. ACM, 2011
S15	Walravens, Nils. "Qualitative indicators for smart city business models: The case of mobile services and applications."Telecommunications Policy 39.3-4 (2015): 218-240.
S16	Seidenstricker, Sven, Erwin Rauch, and Cinzia Battistella. "Business model engineering for distributed manufacturing systems."Procedia CIRP 62 (2017): 135-140.

Continues on next page

Table 2.2 – Continuation from previous page

Id	Paper
S17	Ongena, Guido, Erik Huizer, and Lidwien van de Wijngaert. "Threats and opportunities for new audiovisual cultural heritage archive services: The Dutch case." <i>Telematics and informatics</i> 29.2 (2012): 156-165.
S18	Kolsch, P., et al. "A novel concept for the development of availability-oriented business models." <i>Procedia CIRP</i> 64 (2017): 340-344.

2.2.3 Reporting: answering the research questions

This section discusses the results found in the 18 selected primary studies and offers an analysis of the extracted data with respect to, each research question. Table 2.3 describes a synthesis of the data extracted from the 18 selected studies.

Table 2.3: Synthesis of the data extracted on business modeling.

Id	Method's name	Source discipline	Main concepts	Main goal	Support tool	Graphical notation
S1	Activity system map	Business strategy	Strategic theme and activity	Facilitates the understanding and creation of business strategies	No	Yes
S2	Business models for e-government (BMeG)	E-government	Partner, object, and object ex-change, (dis)advantage	Supports the planning of business models for eGovernment services	Yes	No
S3	Business Model Ontology (BMO)	E-business	Interrelated building blocks, value, value configuration	Proposes a conceptual model of business models	Yes	No
S4	Causal loop diagram	Business strategy	Choice and consequence	Proposes a separation between tactics and strategy using causality theory	No	Yes
S5	e3value	Information systems	Actor, market segment, value, and value ex-change	Develops an e-commerce intensive information system	Yes	Yes
S6	E-business model schematics	E-business	Actor, value, flow, and relation	Migrates from business to e-business	No	Yes

Continues on next page

Table 2.3 – Continuation from previous page

Id	Method's name	Source discipline	Main concepts	Main goal	Support tool	Graphical notation
S7	Eriksson-Penker business extensions of the Unified Modeling Language	Information systems	Processes, events, goals, resources, and rules	Identifies all the use cases (or the correct ones) that best support the business in which the system operates	Yes	Yes
S8	Resource-Event-Agent (REA)	Business accounting	Resource, event, agent, and economic contract	Proposes a generalized accounting framework designed to be used in a shared data environment	Yes	Yes
S9	Strategic Business Model Ontology (SBMO)	Business strategy	Actor and goal	Helps understanding and analyzing the goals, intentions, roles, and the rationale behind the strategic actions in a business environment	Yes	No
S10	Value map	Business strategy	Actor, value, and value exchange	Analyzes the flow of values (tangibles and intangibles)	No	Yes
S11	Value net	Business strategy	Actor, activity, and flow	Helps creating a brand new perspective in strategic analysis (analysis of new strategies from the current ones)	No	No
S12	Value stream map	Business strategy	Actor and value stream	Maps the value streams between the actors to understand the business	No	Yes
S13	Software ECOsystem (SECO)	Ecosystems	Actors and flow	Facilitates further understanding of the business models and value chains of the software ecosystem	No	Yes
S14	eTrade	E-government	Dealer, distributor, wholesaler, retailer, and citizen	Presents the concept of the eTrade Distribution Network as a model for improving to access the eGovernment services and products in a developing country	Yes	No
S15	N/A	E-government	Control and value	Analyzes business models that involve public actors, and municipal governments in particular, in the value network	No	No

Continues on next page

2.2. A MAPPING STUDY ON BUSINESS MODELS

Table 2.3 – Continuation from previous page

Id	Method's name	Source discipline	Main concepts	Main goal	Support tool	Graphical notation
S16	Bussiness Model Engineering	Business strategy	Product/service, gains, pains, and time	Achieves production excellence in each production unit and ensures strategic probability to enhance implemented distributed manufacturing systems	No	No
S17	STOF	Business strategy	Market segment, functionality, cost structure, profit potential, and structure of value network	Analyzes the business-to-consumer market for digital audiovisual services	No	No
S18	N/A	Product-Service System (PSS) ¹	Monetary flow, communication, data, and output	Develops availability-oriented business models	No	Yes

RQ1: For which research area has the method been created? Eight out of 18 studies (44.4%) were created to solve some problem related to business strategy [S1, S4, S9, S10, S11, S12, S16, S17]. The second discipline with more articles is e-government with 16.6% (3 out of 18 studies) [S2, S14, S15] followed by e-business [S3, S6] and information systems [S5, S7] with 11.1% each (2 out of 18). The disciplines with less studies are business accounting [S8], ecosystems [S13], and Product-service system [S18] with 5.5 % each (1 out of 18 studies each).

RQ2: What are its main concepts? Mostly, the studies share a set of common concepts (even if sometimes with different names), such as actors (partner, agent, dealer, distributor, wholesaler, retailer, or citizen) [S2, S5, S6, S7, S8, S9, S10, S11, S12, S13], resources (value, object, product, service, or data) [S2, S4, S5, S6, S8, S10, S12, S15, S16, S17, S18], and transfer of resources (value exchange, object exchange, event, value flow, flow, value stream, or monetary flow) between actors [S2, S5, S6, S8, S10, S11, S12, S13, S18]. Regarding the differences that characterize them, we emphasize, *value configuration* offered by [BMO](#), which is a macro-process necessary to create value for the customer, and *economic contract* offered by [REA](#), which is an aggregation of economic commitments, and *market segment* offered by e3value, which is a group of similar actors.

¹PSS are business models to provide a cohesive delivery of products and services.

RQ3: What is its main goal? For a better structuring of the response, we will aggregate the goals of the studies according to the research areas from which they were created. The studies focusing on business strategies are intended to facilitate understanding [S1, S4, S9, S10, S12] and business analysis [S1, S9, S11, S16, S17] to improve or ensure those business strategies. Regarding the studies focusing on e-government, they are intended to support [S2], analyze [S15] and improve [S14] government services and products. The e-business studies focus on business migration to e-business [S6] and try to standardize e-business models with the creation of conceptual models [S3]. The studies from the information systems discipline focus on the requirements engineering phase of software development, more specifically, on the creation of a requirements specification to develop information systems to better support the business [S5, S7]. The other studies focus on better understanding business accounting [S8] and ecosystems [S13], as well as the development of availability-oriented business models [S18].

RQ4: Is there a supporting tool? Most of the studies do not report any type of tooling support (11 out of 18 studies, corresponding to 61.1%) [S1, S4, S6, S10, S11, S12, S13, S15, S16, S17, S18]. The remaining 7 studies (38.9%) offer tools to support the construction of their methods [S2, S3, S5, S7, S8, S9, S14]. It is important to highlight that the two studies of information systems have a tool to create a model of business value [S5, S7].

RQ5: Does the method use a graphical or textual notation? 10 out of the 18 studies (55.5%) analyzed use a graphical notation to represent their main concepts in a business model [S1, S4, S5, S6, S7, S8, S10, S12, S13, S18]. In contrast, The remaining studies (44.5%) use only textual notation [S2, S3, S9, S11, S14, S15, S16, S17]. Although there is a difference in the number of studies that use a graphic notation and a textual notation, this difference is not statistically significant. Considering the two studies of information systems, both use a graphical notation to represent the concepts of values. However, [S7] uses a more technology-centric approach to creating value models (it creates a [Unified Modeling Language \(UML\)](#) profile, making it less attractive to the business community [81]). Conversely, [S5] uses a more business-centric approach (creating a new model but without the rigor required for a software requirements specification).

2.2.4 Threats to validity

Internal validity. In all systematic mapping studies there is the risk of not including relevant studies. The terms used the in search string of this study were reviewed by external reviewers as recommended in [149]. Interpretation problems can also happen during **planning and execution** due to being totally immersed in the work. To mitigate this, we used Fabbri's best practices checklist [87] to check our work². Also, one external reviewer was involved to check, for a subset of papers, if there were interpretation

²This checklist has a number of questions to evaluate the search string, research protocol, initial selection of studies, final selection of studies, and data extraction.

problems (re-test evaluation method [149]).

External validity: We could have **missed venues** (e.g., conferences and journals) with relevant published works. To avoid this issue, we did not restrict our searchers to venues where more work related to our research was found. We searched in three major digital libraries for relevant related work, and we consider that these libraries are sufficient for our study.

2.3 Two Business modeling approaches

From the list of 18 approaches found in the previous study, we selected two because they were created to facilitate the alignment between the business level and information systems. These two approaches, *Eriksson-Penker business extensions (EPBE)* and *e3value*, are summarized next.

2.3.1 Eriksson-Penker UML business extensions

EPBE is an extension of the *UML* [114]. It was designed to enable the use of *UML* in business modeling. According to the authors, the main motivation of this method is to describe a business in terms of processes that satisfy objectives through the collaboration of different types of resources [114]. However, as already discussed in Section 2.1.2, the goal of a business process model is different from that of a business model [106]. Rules define conditions and constraints on how processes and resources should relate to and how they should behave [114]. All of this can be mapped into objects, relationships, and interactions between objects [114]. In this approach, the five fundamental elements used to describe a business are [114]:

- **Resources:** is everything the company uses, consumes or produces: people, materials, information and products. Resources are manipulated and managed through processes, and are classified as: physical, abstract and information. Resources are represented in *UML* object diagrams.
- **Processes:** are the activities carried out by the business. They describe how the work runs in the enterprise, and are bounded by rules. Processes are represented in *UML* activity diagrams.
- **Rules:** are the definitions or restrictions of some aspect of the business. Rules determine how a business should be managed or how resources should be structured and used. They can be created by the company itself or imposed by external entities (government, associations, unions, etc.). Rules are represented in *UML* class diagrams.

- Goals: represent the reason of the company, or the results that the business expects to achieve. The set of high-level objectives forms the company's strategy. Objectives can be divided and distributed among the various processes of the company. Objectives are represented in UML class diagrams.
- Events: are actions that occur in the external world and that affect processes. Events are represented in UML class diagrams.

The authors highlight that their method provides a foundation to support the creation of information systems. However, they at no time describe an approach to modeling the company's value proposition and its operational model. Figure 2.4 shows the EPBE metamodel.

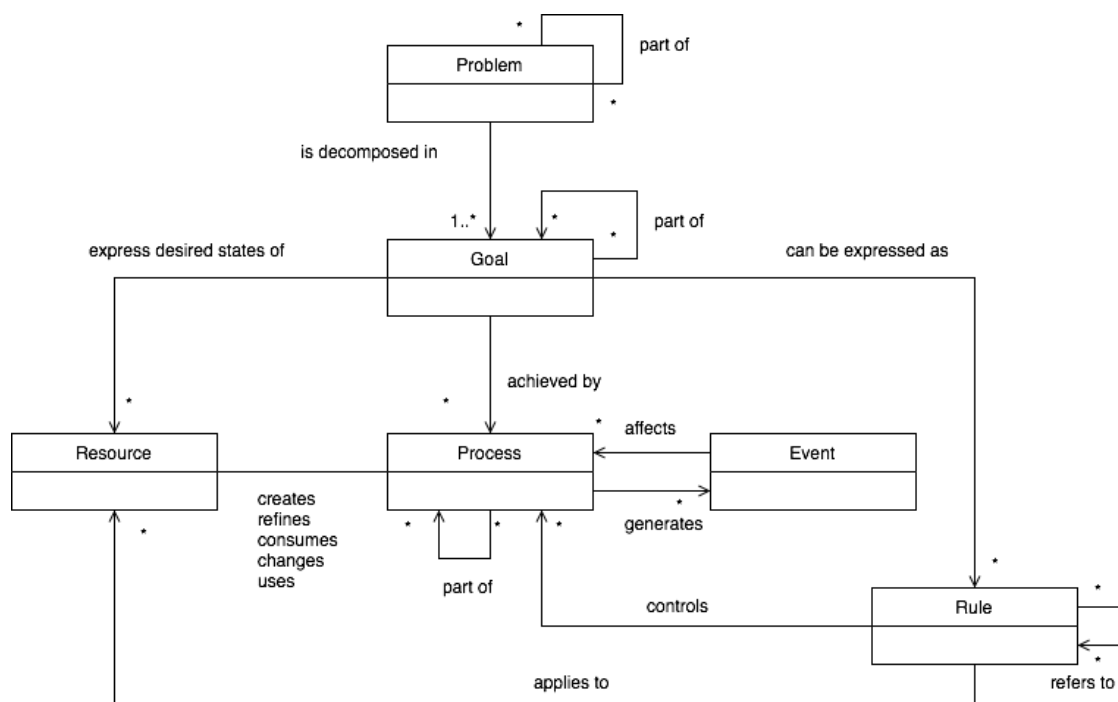


Figure 2.4: EPBE metamodel.

The metamodel shows how processes attempt to achieve goals. A goal is established to overcome one or more problems and expresses the desired state of one or more resources. Goals can be expressed as rules that control the process. A process interacts with resources through an interface and can cause the states of resources to change. A process also interacts with other processes by generating or handling events. Resources can be physical (e.g., people), abstract (e.g., invoice), or information, holding information about another resource (e.g., database record).

2.3.2 e3value

The e3value method offers modeling constructs for representing graphically and analyzing business requirements from an economic point of view [105]. The method is

composed of fifty concepts, and the main ones are [104]:

- Elementary actor: is an economically independent (and often also legal) entity. An elementary actor does not contain value interfaces of other actors.
- Composite actor: is a specialized actor grouping value interfaces of other actors.
- Market segment: is a concept that breaks a market (consisting of actors) into segments that share common properties.
- Value interface: consists of one value transferring (in-going offering or out-going offering).
- Value transfer: shows what an actor offers to (an out-going offering) or requests from (an in-going offering) his/her environment, and closely relates to the value interface concept.
- Value port: is used to interconnect actors so that they are able to exchange value objects.
- Value object: is a service, a product, or even an experience, which is of economic value for at least one of the actors involved in a value model.
- Value exchange: represents one or more potential trades of value object instances between value ports.
- Value transaction: aggregates all value exchanges.
- Value activity: is a collection of operational activities which can be assigned as a whole to actors.
- Start stimulus (customer needs): is the beginning of a value scenario. This concept is inherited from the [Use Case Maps \(UCM\)](#) [48].
- Stop stimulus (scenario boundary): is the end of a value scenario. This concept is inherited from the [UCM](#) [48].
- AND element: is a logical operator used to split or collapse paths of value scenarios. This concept is inherited from the [UCM](#) [48].
- OR element: is a logical operator used to split or collapse paths of value scenarios. This concept is inherited from the [UCM](#) [48].
- Connect element: is used to link the graphical elements of the model to show a complete path of value scenario. This concept is inherited from the [UCM](#) [48].

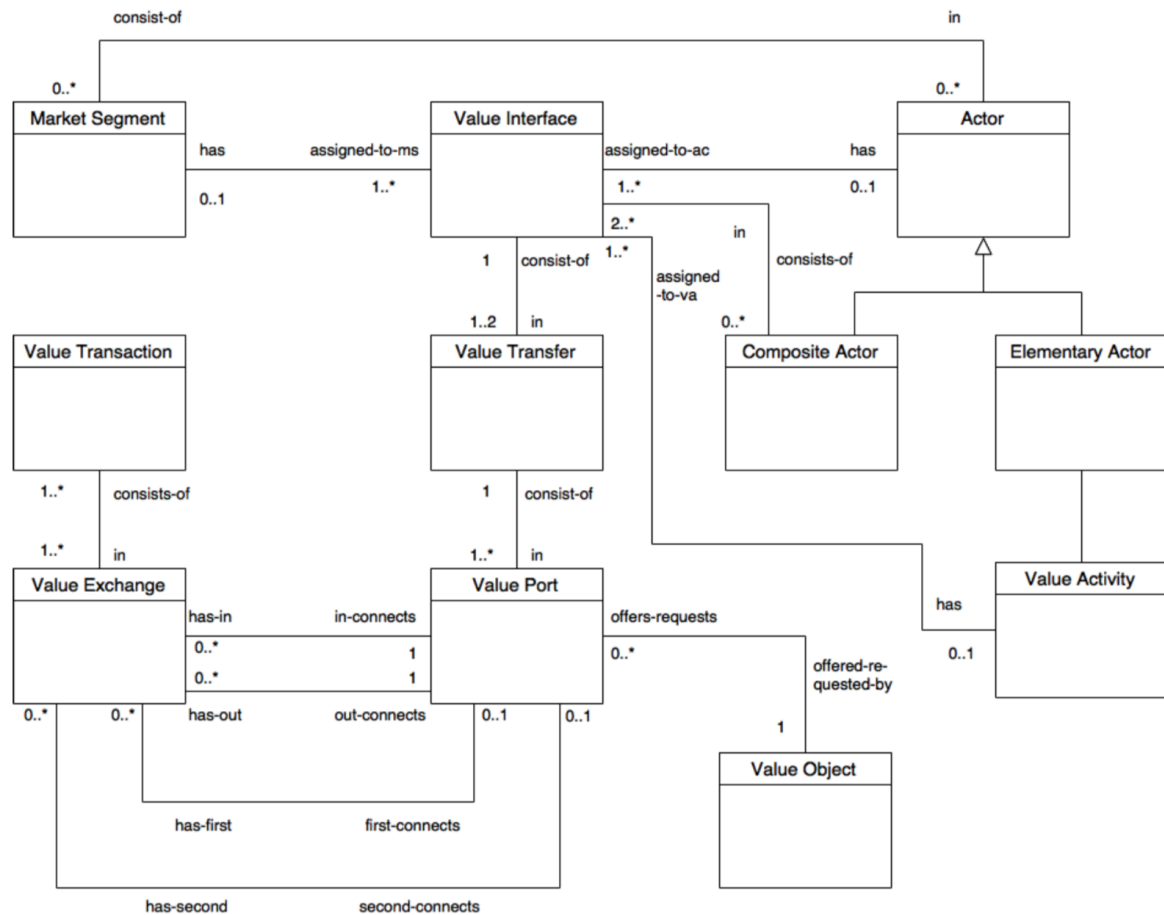


Figure 2.5: e3value metamodel, taken from [104].

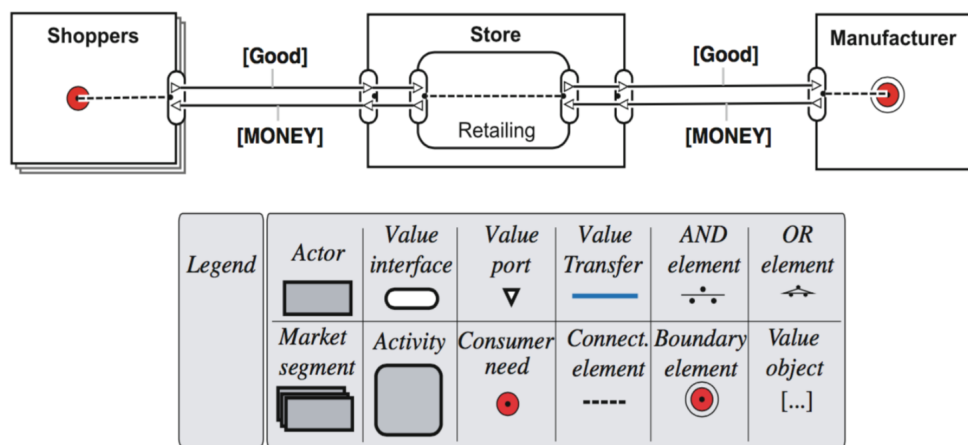


Figure 2.6: e3value example extracted from [206].

Figure 2.5 presents the e3value metamodel (note that some of these concepts are not present [103, 212]), and Figure 2.6 exemplifies an e3value model.

In order to represent value exchange scenarios, the e3value model inherited the start stimulus, the stop stimulus, the AND and OR logic operators, and the connect element

from UCM³ [48]. Although these elements are contained in the e3value model (see Figure 2.6), they are absent from the metamodel (see Figure 2.5), showing that the e3value metamodel is incomplete.

The start stimulus represents customer needs, that is, the beginning of a value scenario, and the stop stimulus represents the end of a value scenario. A connection element links a start-stop stimulus to a value interface or links value interfaces of the same actor internally. As a lot of value scenarios are represented in a unique e3value model, AND and OR elements are used to split or collapse paths of value scenarios, reusing start and stop stimulus elements.

2.4 Final considerations

This Chapter presents a state of the art on business modeling, delimiting the scope of our work and clarifying the main concepts of the area. As part of the task, we performed a systematic mapping study to analyze a complete list of the business modeling methods published in the literature. Most of the studies found were created to solve problems related to business strategy. The various studies share a set of common concepts (such as actors, value, and value exchange) aiming at facilitating business understanding and analysis to improve business strategies. Most studies do not offer supporting tools, what impairs negatively to the use of their methods. Also, most studies focus on graphical notations to model business concepts. However, the difference between who uses and who does not use graphical notations is not significant (two more studies favor graphical notations). From the list of 18 approaches studied, two (EPBE and e3value) aimed at facilitating the alignment between business and information systems. For this reason, they were summarized. On the one hand, EPBE does not at any time describe an approach to modeling the company's value proposition and, on the other hand, e3value is difficult to be used in a systematic software development due to some ambiguous concept representation and also because the same model represents static and dynamic perspectives of the problem/solution, a little against the good practice of separation of concerns. Because of this, we decided to focus our research on creating a new easy-to-use business modeling method that is useful for both business and software engineering professionals. In addition, we used e3value as the basis of comparison through a family of three controlled experiments.

³Use Case Maps is a requirements language which have the notion of path to show how a particular scenario works.

SOFTWARE ARCHITECTURE

The Software Architecture research community has accumulated a large body of knowledge over decades, to create and evolve new techniques, tools, processes, methods, and frameworks. However, researchers and practitioners may have a hard time locating consolidated evidence on this [Software architecture](#) body of knowledge, as the available information is disseminated in several different publications. This chapter offers an overview of software architecture. It starts with a brief history of the origin of the software architecture and defining the main concepts needed to a broad understanding of the topic. Next, it describes the principles and concepts on Service-Oriented Architecture (SOA) — a widely used business practice to develop enterprise information systems [136]. It then gives a brief description of model-driven engineering aiming at providing a background on this methodology created to build complex software systems. Finally, it describes the conduction of a systematic tertiary study, aggregating consolidated findings on software architecture and facilitating the work of researchers and practitioners in learning about the coverage and main results of existing work. For example, we identified the nonexistence of secondary studies looking for software architecture derivation methods. These and other relatively unexplored niches of research were synthesized to answer the research questions of the tertiary study.

3.1 What is software architecture?

This section describes a brief history of the origin of Software Architecture and defines some concepts for a broad understanding of the topic.

Origin. According to Kruchten *et al.* [160], the first reference to the term *software architecture* occurred in 1969 in a conference on software engineering techniques organized by

[North Atlantic Treaty Organization \(NATO\)](#) [50]. From then until the late 1980s, the word “architecture” was used mostly in the sense of system architecture, meaning a computer system’s physical structure [160]. Software architecture as a distinct discipline started to emerge in the 1990’s [160], and in 2001 was created the first Working IEEE/IFIP Conference on Software Architecture [209]. Ever since then, Software Architecture has been largely researched by industry and academia.

Software architecture definition. The term software architecture is widely used in various contexts within Software Engineering [120], which reflects the growth or recognition of its importance. However, the variety of contexts in which the term is used is an indication that the concept is still is not very well defined [159]. Among the various software architecture definitions we selected the following ones in Table 3.1.

Table 3.1: Software architecture definitions.

Authors	Reference	Definition
Perry and Wolf	[201]	Software architecture represents the organization of collections of interconnected components that obey certain constraints on the form of interaction.
Shaw and Garlan	[172]	Software architecture is defined as a representation of the system in terms of computational components and their relationships.
Hofmeister <i>et al.</i>	[119]	Software architecture is the bridge between the requirements of the system and its implementation, serving as a guide for all activities of the software development process.
Jazayeri <i>et al.</i>	[133]	Software architecture assists in the management of software complexity, being the satisfaction of the functional and non-functional requirements of a software system.
Putman	[210]	Software architecture consists of well-defined rules and concepts related to all aspects of the system, from functional requirements to non-functional and semantic behavior of the system.
Garlan	[99]	Software architecture plays a fundamental role in the following aspects of the development process: analysis, common understanding, reuse, construction, evolution and maintenance, and management.

Continues on next page

Table 3.1 – *Continuation from previous page*

Authors	Reference	Definition
Kruchten	[159]	Software architecture should consider the following aspects: The organization of a software system; The selection of the structural elements and their interfaces; and The Architectural style that guides the organization of the elements and their interfaces.
Sommerville	[235]	Software architecture represents the fundamental framework for structuring a software system. Attributes of a system such as performance, security and availability are influenced by the software architecture used.
Bass, Clements, and Kazman	[26]	Software architecture of a computational system is the “structure of structures”, which encompasses components, their external properties and their relationships, creating a system abstraction that suppresses details of components that do not affect form how they are used.
ISO/IEC/IEEE 42010:2011	[127]	Software architecture covers the fundamental concepts or properties of a system incorporated in its elements, relationships and principles of design and evolution

An analysis of the presented definitions, despite the differences [120], indicates that, at its core, Software Architecture refers to the software elements, their relationships with each other and its environment [26, 172, 201, 235], the principles governing its design and evolution [99, 119, 127, 133, 159, 210], as well as the quality properties these elements support [24]. This definition embodies the two perspectives, where the software architecture discipline guides the development, and organization of structural choices and determines how the system should be constructed and evolved [89]. Since the 1990s, software architecture has received a lot of attention within the area of Software Engineering due to its criticality as one of the most important success factors in computer systems projects [99].

Software reference architecture. A software architecture serving as a baseline for the design activities that are executed in a software development project. It also serves as an anticipated manifestation of the project decisions, influencing factors such as development time, cost and maintenance, the definition of the constraints of implementation, emphasizing the quality attributes that the system must have to meet the business needs [26]. As a baseline, a software reference architecture allows stakeholders (end-users,

developers, test engineers, project managers, etc.) having a high-level understanding of the software structure, enabling the communication among them, and facilitating the analysis and validation of the system [24, 235]. The larger and more complex the system is, the greater the need for easier communication, which directly impacts on the success of a software project [24, 235]. A software reference architecture can also be used as a planning and management tool, assisting system managers in decision making and minimizing risks and uncertainties in software projects [199]. In this sense, **Software reference architecture** models have been used to facilitate and guide the architectural decision-making, for instance in the choice of architectural patterns [172] and styles [91] during the architectural design process [188, 235] (some information on architectural styles and patterns is given next). Indeed, a *software reference architecture* model offers a template solution of an architecture for a particular domain, expressing ways of organizing the fundamental structure of a system through high-level reusable solutions [24, 235].

Architectural viewpoints. The description of a software architecture is directly related to the use of abstraction techniques. The use of abstractions simplifies the problem, since it provides separation of concerns and responsibilities that facilitate the development of software systems [170]. A software architecture can be described in different levels of abstraction and using different views with the same level of abstraction. The purpose for using points of view stems from the fact that different stakeholders have different interests in the system [19]. For example, Kruchten created the 4 + 1 views approach to detail five different, but complementary, viewpoints to represent a software architecture [161]. In Kruchten's approach, system engineers use a *physical view* to describe the mapping of the software onto the hardware, and a *process view* to capture the concurrency and synchronization aspects of the design. Software engineers and data specialists use a *logical view* to create the object model of the design (when an object-oriented design methodology is used). Project managers and the software configuration team use the *development view* to describe the static organization of the software in its development environment. Finally, all these four views are created from the architecturally relevant requirements which are illustrated in the *scenario view*.

Architectural drivers. The transition between the requirements of a problem and its solution involves architectural decisions, which are at the core of software architecture [43, 131, 158, 255], as they are related to the satisfaction of functional and non-functional requirements¹ [25, 270]. Although it is well accepted that **NFRs** are difficult to describe, satisfy and track in a software architecture specification [56, 110], it is also well accepted that they are the main drivers of the architectural decision process [25, 89, 125].

¹Functional requirements describe what the system does, and **NFRs** typically refer to the operational quality of a system, as well as the constraints imposed on a solution [58].

Architectural description. The representation of the system solution for a specific point of view is carried out through languages and notations. These languages and notations are called [Architectural Description Language \(ADL\)s](#). An ADL provides both a conceptual framework and a concrete syntax for characterizing software architectures through a model [194].

Architectural styles and patterns. Styles are mechanisms for categorizing architectures and for defining their common characteristics [70]. Each style provides a high-level abstraction for the interactions of components, capturing the essence of a solution of interaction by ignoring the incidental details of the rest of the architecture [224]. There are several different architectural styles (e.g., layered architecture [220], pipe-filter [1], and SOA [136]). Due to the large number of existing architectural styles, their properties and benefits have been systematically described, for example, the catalog in [26]. On the other hand, the term architectural pattern has been used to describe the same concept of architectural style [26] although the meaning is somewhat different. For example, a layered architecture is an architectural style where the software components are organized into horizontal layers, each layer performing a specific role within the application [220]. This style does not specify the number of layers needed to structure the software, in contrast, it only defines the logic of this structuring. However, some architectural patterns use the knowledge offered by the layered style, providing more practical and specific details regarding, for example, the number of layers, abstraction level and even how to implement it in different programming languages. According to Bass *et al.* [26], a *Pattern* is a context-problem-solution triple while a *Style* is a condensation that focuses most heavily on the solution part.

3.2 Service-oriented architecture

Service-Oriented Architecture (SOA) is an [Architectural style](#) to support the development of distributed applications, even in heterogeneous environments [136, 196]. As SOA is a widely used business practice to develop enterprise information systems [136], we will be using it to facilitate the alignment between business and IT solutions. Next, we offer an overview of SOA's main concepts and infrastructure, highlighting its main strengths and weaknesses, as well as [SoaML](#), a language for describing systems structured using [Service-Oriented Architecture](#).

Service. A *service*, the SOA core concept, is defined as an autonomous, minimally coupled, and platform-independent entity that can be published and used in novel ways [195]. Therefore, each service encapsulates in different contexts. Although services exist autonomously and independently, they are not isolated from each other [84]. This is aligned with Erl's view [85], where services are defined as physically independent software programs with distinct characteristics that support the achievement of the strategic

objectives associated with service-oriented computing. In this context, a service can be defined as a self-contained module that provides some concrete functionality to its environment. Although independent, a service is typically built to relate to, or communicate with, other services that are available in any location. According to the [OMG](#), a service is the delivery of value to another party, enabled by one or more capabilities [189]. Each service is created in a distinct context, having a set of capabilities and sometimes some preconditions to be invoked. For a service, any other service is a potential candidate to serve as a partner.

Services classification. There are multiple ways to classify services [136]. In the context of information system design, the services can be classified as business services, informational services, and utility services [71, 262]. *Business services* refer to economic services provided by an economic actor to fulfill a customer need. *Informational services* referring to software services aiming at producing information or enhancing communication. *Business services* can use groups of *information services* to achieve their purpose. And, finally, *utility services*, strictly related to the technological platform supporting the information services, provide functions for data storage or execution of programs.

Infrastructure. Some standards based on [Extensible Markup Language \(XML\)](#) protocols [53], such as [Simple Object Access Protocol \(SOAP\)](#) and [Web Services Description Language \(WSDL\)](#), are designed to support service communication and information exchange [85]. Thanks to these standards, the services are independent from the platform and implementation language. The software can be constructed by composing a set of services. A service can be local or external (provided by different suppliers) [85]. Some [SOA](#) infrastructure definitions include the term Web Services, but [SOA](#) is not the same as Web Services. Indeed, while [SOA](#) is an architectural style, web Services are one possible way to realize the infrastructure by using a specific implementation strategy [136].

Figure 3.1 encapsulates the idea of a [SOA](#) infrastructure, where a *Services provider* designs, implements, and specifies *service*-based software. Then a *services provider publishes* information about these services in a *service record* accessible to their customers (also known as *service requesters* or *Service consumers*). *Service consumers* that wish to use a service, use a mechanism to *discover* the specification of that service and locate the service provider. They can then associate their application with this specific service and communicate with it , using of service communication protocols (*Join(SOAP)*) [235].

Advantages and weakness. Given that interactions occur with loosely coupled services that operate independently [136, 195, 196], [SOA](#) is widely used for the development of [IS](#). [SOA](#) facilitates the system's evolution to reflect business changes, improving the alignment between the developed software and the needs of the business [20, 136, 196]. In summary, the advantages of [SOA](#) are [168]:

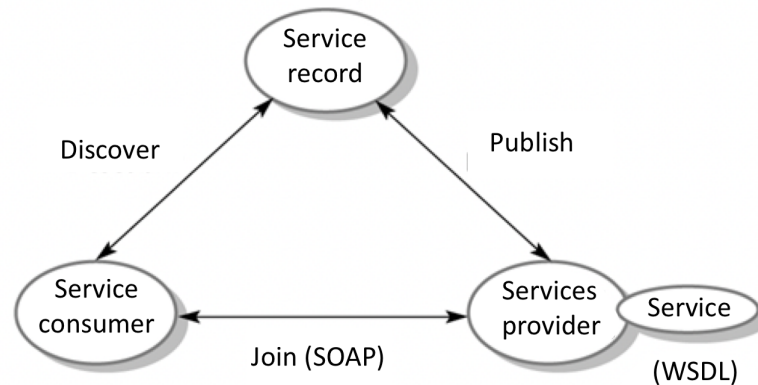


Figure 3.1: General view of a Service-Oriented Architecture [235].

- Ease of integration and interoperability between services, what increases efficiency in the use of available resources.
- Better scalability and ability to develop applications independently and in parallel.
- Reduction in system development costs due to component reuse.

Regarding the SOA weakness, the critical aspects of developing an efficient SOA solution are:

- The identification of services from the requirements and their mapping to an appropriate SOA reference architecture has not been done satisfactorily [7, 18]
- Lack of approaches using *business value* models to align the business needs and the software services (most methods use business process models rather than *business value* models [20, 113]). This topic is still a challenge, requiring the cooperation between the business and the IT communities [7, 113],
- Identification, monitoring, and management of NFRs still need to be addressed systematically by the software architecture community [7, 113].

SoaML. According to the OMG, SOA is an architectural paradigm for defining how people, organizations, and systems of services delivery act to achieve results [31]. In this context, SoaML [189] is an OMG standard that aims to design and model SOA solutions using UML. It is a set of UML extensions to support SOA modeling, providing a SOA abstraction focused on the description of participants' needs and resources [72], modeling both business and IT perspectives. The elements made available by SoaML allow the creation of UML diagrams to represent the structures of software systems in SOA, allowing the generation of artifacts² for the implementation of software architectures through the existing support tools.

²In general, the artifacts are generated using model-driven engineering techniques. Section 1.5 describe about model-driven engineering.

A characteristic that stands out in *SoaML* is its reach. Several technology-specific tools allow modeling services using *SoaML*. However, such tools do not facilitate describing a high-level service-oriented architecture in a complete way, that is, describing how services and service participants work together to deliver business value [262]. This is a limitation of these tools in supporting *SOA*, and it is known that organizations are most effective when they understand their technology services and how they relate to their business services [34, 262].

Figure 3.2 illustrates an example of a *SoaML* capability model. *SoaML* capability represents the ability to act and produce a result that achieves a specific goal. For example, Figure 3.2 shows that for an online store to be able to offer a *Purchase order*, it also needs to offer a *shopping cart*, a *purchase request* and a *delivery* service. In this way, a capability is presented as a cohesive set of functions or resources that a participant provider offers through of a service [31]. Capabilities are specified without regard to how a particular service should be implemented and subsequently offered to service consumers from a service provider. Through capability model, developers can analyze how services will relate and how they can be combined to form a larger capability before being assigned to a particular participant [189]. A capability can be represented by a class diagram or component diagram with the «Capability» stereotype.

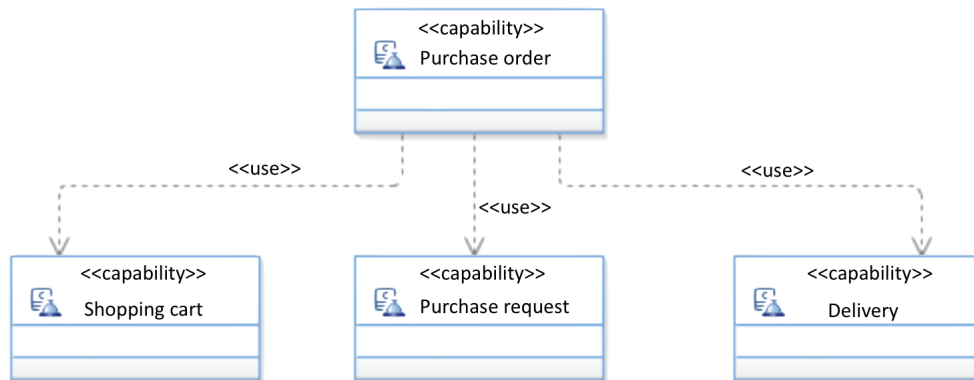


Figure 3.2: Example of a *SoaML* capability model for an order handling system.

Figure 3.3-a presents an ecore metamodel of the service architecture model (or *SoaML* Component Diagram) to design the collaboration of service contracts in *SoaML*. A *Participant* can be a service consumer or a service provider (*ParticipantType*) of a *ServiceContract*. In *SOA*, *NFRs* refer to the quality or capability that satisfies customer specifications and they are explicitly demanded by service consumers and can be formalized by Service Level Agreement [113]. A pure *SoaML* service architecture model does not represent the quality of services concepts, reason why *OMG* adopted the QoS&FT profile [190]. Part of the conceptual representation of this profile is marked in grey in Figure 3.3-a. Figure 3.3-b shows an example of a *SoaML* specification with a description of its concepts, showing how participants (represented as rectangles) work together for a purpose by providing or

using services. A service is expressed in the model as a service-Contract (dashed circle). In this example, an *Applicant* requests a Retirement Insurance Benefit (RIB) to a *Handler*. The *Handler* checks the Social Security number (SSN) of the *Applicant* with a *SSN Matcher* and, depending on the result, it applies for RIB claim to be processed by the *RIB claim processor*.

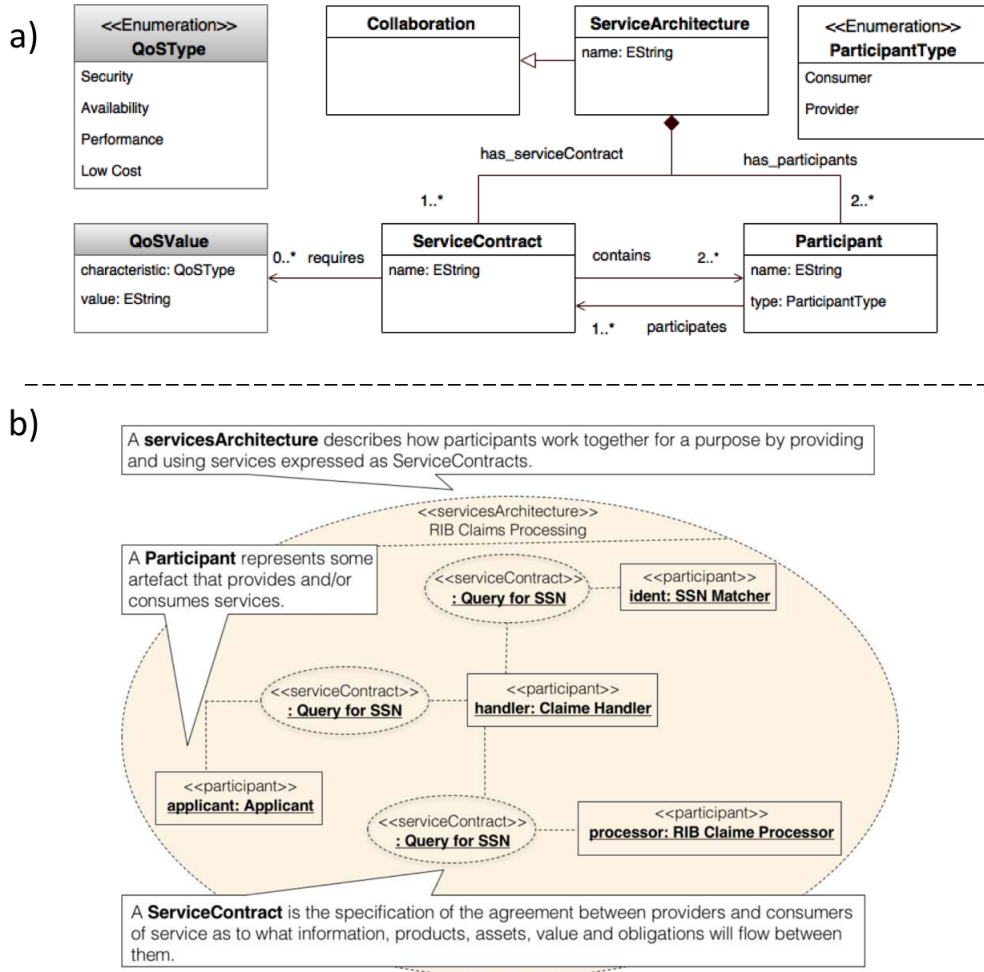


Figure 3.3: a) Ecore SoaML ServicesArchitecture metamodel from [189] extended with part of the QoS& FT profile, and b) SoaML specification, taken from [77].

3.3 State of the art on software architecture: an Evidence-Based Tertiary Study

The software architecture area is too broad for a single study, but there are already several secondary studies, in the form of literature reviews that partially aggregate information in the area. As a consequence, we leverage those secondary studies, by conducting a tertiary study to provide a consolidated state of the art and practices in software architecture. Our main objective in conducting this tertiary study was to verify the existence

of an updated state of the art about the derivation methods of an architecture model from a requirements specification but not only that. Thus, the tertiary study aims at the following contributions:

- A mapping of the most representative secondary studies that cover software architecture and their origin. This mapping is expected to serve as a starting point for researchers and practitioners interested in software architecture to locate relevant consolidated reviews in the area, and the experts responsible for those reviews.
- An annotated overview of the existing aggregated information on software architecture. By describing the main contributions of the included secondary studies, we provide a unique resource from which researchers and practitioners can start exploring future work.
- A report on the level of consolidation of the aggregated information on software architecture. One of the purposes that a literature review can play is to aggregate and consolidate information that would be otherwise scattered in several different publications, while conserving traceability links to the original research sources. In this tertiary review, we are interested in knowing to what extent this was achieved in the reported literature reviews.

This study is a work in [EBSE](#) [151]. [EBSE](#) aims at collecting the best available evidence to address software engineering research questions created by both practitioners and researchers. Typically, this is performed by aggregating existing empirical studies on a particular topic and conducting a literature review on them, in an impartial way. Our study was conducted following the guidelines on [SS](#), detailed in [148], of which the main phases are planning, conducting, and reporting (as detailed in Section 2.2). Our goal, with this decision, was to gather a broader overview of the current state of the art and practice of software architecture. The next sections discuss how those three phases were performed.

3.3.1 Planning: Defining the protocol

The protocol shows the formulation of the research questions, search strategies, search string and research sources, the description of the process followed to search, classify and assess for quality of the identified secondary studies, the design of data extraction procedure, and, finally, the development and evaluation of the review protocol.

Formulating the research questions. We used the [PICOC](#) method [205] to structure and formulate our research questions. PICOC is a method used to describe the five elements of a searchable question. “PICOC” is an acronym that stands for **P**opulation (who?), **I**ntervention (What or How?), **C**omparison (Compared to what?), **O**utcome (What

3.3. STATE OF THE ART ON SOFTWARE ARCHITECTURE: AN EVIDENCE-BASED TERTIARY STUDY

are you trying to accomplish / improve?), and Context (In what kind of organization / circumstances?). Table 3.2 shows the PICOC analysis of tertiary study.

Table 3.2: **PICOC** Analysis for the tertiary study about secondary studies on Software Architecture.

Population	General empirical studies on Software architecture created following the best practices of EBSE . In other words, we want systematic studies with any topic, system or domain of application in the area of software architecture.
Intervention	The empirical studies must be secondary studies (i.e., systematic literature reviews or mapping studies), as they follow a rigorous and systematic procedure for search and selection of the sample primary studies to review, hence leading to more complete results. Our goal is to characterize the state of the art and practice in software architecture through the currently available information extracted from the secondary studies and characterize the community in this research domain.
Comparison	Not applicable, as our intention is to classify the topics of the secondary studies from their research questions.
Outcome	The main outcome is (i) provide the available information concerning software architecture and (ii) identify the key players in consolidating knowledge on the research area. This include to check the quantity of primary studies that were included in these reviews, the time span covered by these reviews, if the reviews provide the list of the included primary studies, if the quality of the primary studies were assessed, identify the target audience of these reviews, identify the quantity of systematic studies are available, the origin of these reviews, and what is the impact of these reviews, in terms of citations.
Context	Research papers. We are working in a research context with experts of the domain as well as other practitioners, academics, consultants and students with published research results.

The **PICOC** Analysis led to the definition of the following three research questions:

RQ1: What is the software architecture information systematically aggregated by existing secondary studies currently available? Our goal is to characterize the current state of the art in this research domain.

RQ2: What is the current status of consolidation of data collected from different literature reviews on software architecture? One of the potentially key benefits to be taken from literature reviews is to be able to aggregate data collected independently in different studies.

RQ3: Who is performing literature surveys? This question is of a demographic nature. Our goal is to characterize the community with interest in consolidating knowledge on software architecture.

The research question RQ1 was broken down into the following six sub-questions:

RQ1.1: How many primary studies are included in these reviews? This provides the consolidation level of the research topic.

RQ1.2: What is the time span covered by these reviews? This may shed some light on whether the area is consolidated.

RQ1.3: Is the list of the included primary studies available? This provide a traceability to the information source.

RQ1.4: Is the quality of the primary studies assessed? This provide an indication of the maturity level of secondary study.

RQ1.5: Who is the secondary study targeted to? The target audience can be made of researchers, practitioners, or both.

RQ1.6: Which publication venues are most commonly used? This shows us which venues are most interested in this type of research.

Similarly, we decomposed RQ2 into the following two sub-questions:

RQ2.1 What are the software architecture research topics that contain systematic secondary studies? This provides a coverage of the software architecture topics.

RQ2.2 Is there some secondary study focusing on methods for the derivation of software architecture models from requirements specifications? In our Ph.D. research, we need understand the context of these type of methods (e.g., goal, application domain, and inputs), the benefits they offer to the users (e.g., benefits and limitations), their content (e.g., architectural viewpoints, architectural description language used, level of automation), and how they validate the software architecture generated (e.g., case study, experimentation, illustrative example). If we do not find a study (or set of studies) that provides us with such information, then we will have to conduct a systematic mapping study to aggregate the data from the primary studies and answer this question.

And the same decomposition approach was followed for RQ3:

RQ3.1: How many systematic reviews, including systematic literature reviews and mapping studies, addressing any topic in software architecture, are available? This provides an overview on the liveliness of the community consolidating information on software architecture.

RQ3.2: What is the origin of these reviews? This covers organizations and countries and is aimed at better understanding the extent to which software architecture is becoming a global concern.

RQ3.3: What is the impact of these reviews, in terms of citations? This will provide some insight on the “popularity” of this research topic.

Defining the search strategy. Our search strategy used two complementary methods [269]: an *automatic search* using the search terms discussed next to discover secondary studies on electronic data sources, and a *manual search* in Google Scholar to look for additional secondary studies on software architecture.

Formulating the search string. The key terms for the search string were derived from the research questions [149] and are listed in Table 3.3. We conducted a pilot to validate the search string, as recommended in [45], adding synonyms of key terms. The search string includes the terms “*systematic review*”, “*literature review*”, “*mapping study*”, “*systematic study*”, “*software architecture*”, “*reference architecture*”, “*architecture design*”, and “*architectural design*”, concatenated with Boolean operators. As different digital libraries use different syntax and limitations for search queries, we formulated dedicated search queries for each digital libraries from the search terms listed in Table 3.3. The search strings used in the digital libraries were structured according to the following logic: (“*systematic review*” OR “*literature review*” OR “*mapping study*” OR “*systematic study*”) AND (“*software architecture*” OR “*reference architecture*” OR “*service architecture*” OR “*architecture design*” OR “*architectural design*”).

Table 3.3: Research Query Building

Systematic Study	“systematic review” OR “literature review” OR “mapping study” OR “systematic study”
Software Architecture	“software architecture” OR “reference architecture” OR “service architecture” OR “architecture design” OR “architectural design”

Selecting research source. The search string was ran in four digital libraries: IEEEExplore [124], ACM Digital Library [5], Science Direct [75], and SpringerLink [246]. These digital libraries were selected because they cover the most important publications in software architecture (e.g., journal papers, conference proceedings and workshop papers). It is important to notice that in any systematic study there is a potential of missing the relevant studies in the selected digital libraries. Because of this, we used a manual search approach on Google Scholar [102] aiming at complementing the results of the automatic searches with additional systematic studies.

Selecting studies. We included all candidate studies returned from the automated search (I1) and the selected candidate studies from the manual search (I2). Exclusion criteria were defined to select the secondary studies for analysis from the set of all candidate studies initially obtained. We read the title and abstract from all candidate studies in order to exclude the papers that were not systematic literature reviews or mapping studies focusing on software architecture (E1), duplicated papers (E2), and papers written in a language different from English (E3). Once the title and abstract analysis was concluded, we joined the all candidate studies to make a complete analysis reading the papers’ full text by applying the same exclusions criteria.

The selected studies were cataloged into a bibliography management tool [197], and the data related to our research was extracted to a spreadsheet workbook previously

structured as a form. We extracted publication data showing characteristics of the included secondary studies (i.e., research source, title, authors, year of publication, type of publication, and venue) and information required to answer our two research questions.

Classifying selected studies. We used three important facets for classifying the studies namely *study type*, *research topics*, and *publication venues*. The *study type* facet consists of the following categories: Systematic Literature Review (SLR) and Systematic Mapping Study (SMS). To do the *research topic* facet, we selected the topics of interest from the special issue “*New Frontiers in Software Architecture*” of the Journal of Systems and Software (JSS) and from the call for papers of the Working IEEE/IFIP Conference on Software Architecture (WICSA), and used them to classify the research topics in Software Architecture. Thus, the *research topic* facet consists of twenty five categories, listed in Table 3.8. The *publication venue* facet consists of journal, conference, symposium, workshop, technical report, and others. These facets are used to classify the studies for answering the research questions.

Assessing the quality of the studies. Although there is no agreed definition on what a high level quality study is, there is a common agreement that the quality of the chosen primary studies is critical for obtaining trustworthy results in empirical studies [149]. We followed the widely accepted quality assessment criteria proposed by Kitchenham *et al.* [150] for systematic tertiary studies. The summary of these quality assessment criteria is as follows:

- QA1:** Are the inclusion and exclusion criteria described and appropriate? Score: YES, if inclusion criteria are explicit, PARTLY, if the inclusion criteria are implicit; NO, if inclusion criteria are not defined.
- QA2:** Is the literature search likely to have covered all relevant studies? Score: YES, if two or more complete search strategies are used. For instance, automated search on minimum four digital libraries and manual search on all potential publication forums considered to be complete; PARTLY if search strategy has complete primary search strategy and incomplete or no secondary search strategy; NO, the authors have used incomplete search strategy. For instance, a search in up two digital libraries or in an extremely restricted set of publication forums is considered to be incomplete.
- QA3:** Did the reviewers assess the quality/validity of the included studies? Score: YES, if there is explicit quality criteria definition, and reporting of the score; PARTLY, if quality issues are covered in the research questions; NO, if there is no explicit quality assessment.
- QA4:** Were the basic data/studies adequately described? Score: YES, if the information about each primary study is reported; PARTLY, if the information about primary

studies are grouped or only the summary is provided; NO, if there is no data about each primary study.

The scoring procedure was YES = 1, PARTLY = 0.5, NO = 0, or Unknown (i.e., the information is not specified).

Collecting the Data. The following data was extracted from selected systematic studies and the collected data is used to map the studies under the appropriated categories:

- **Number of selected primary studies:** Quantity of the selected primary studies reported in the systematic study to answer RQ1.1.
- **Time span:** The date range used to search the primary studies to answer RQ1.2.
- **List of the included primary studies:** Verify if the systematic study provides the list of the selected primary studies to answer RQ1.3 (YES or NO).
- **Quality assessment data:** Assess the quality of the studies in relation to search strategy, inclusion/exclusion criteria, quality assessment strategy, and level of the detail about the primary studies to answer RQ1.4.
- **Target audience:** Find out the target audience to answer RQ1.5 (*Researchers, practitioners, or both*).
- **Publication data:** Find out the publication types, publishers and the publication venues where the selected systematic studies are published to answer RQ1.6.
- **Research topic:** Number of papers per category of the classification scheme to answer RQ2.
- **Number of selected systematic studies:** Quantity of systematic literature review and mapping studies focusing on software architecture to answer RQ3.1.
- **Author details:** Author names, affiliation, country to uniquely identify the studies to answer RQ3.2
- **Impact:** Number of citations in Google scholar³ [102] to answer RQ3.3.

Reviewing the protocol. The protocol was reviewed following Kitchenham's guidelines [148] by three external reviewers to reduce bias (all of them have published systematic studies).

³This includes self citations.

3.3.2 Conduction

This section presents the search results, identifies relevant systematic studies to answer the ten research questions, and shows the quality assessment scores. Figure 3.4 shows the process performed to search for the secondary studies.

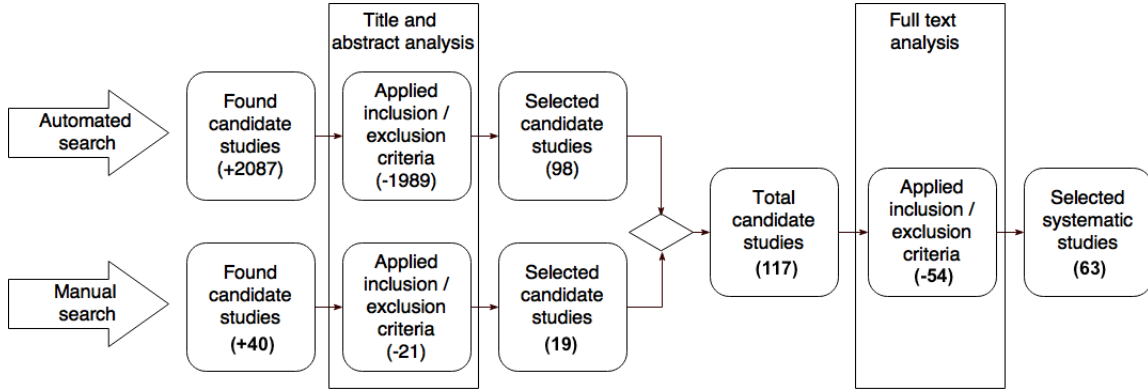


Figure 3.4: Study searching and selection process.

Results. We performed the automated search in four digital libraries (IEEEExplore [124], ACM [5], ScienceDirect [75], and Springerlink [246]) and the manual search in Google Scholar [102]. We obtained a set of 2127 (2087 from automated search + 40 from manual search) candidate secondary studies. The subsequent filtering process (as detailed in Table 3.4) resulted in 63 secondary studies. The list of selected systematic studies is in Table 3.5.

The last row of Table 3.4 shows the total number of included secondary studies from the digital libraries (automatic and manual searches). Despite the apparent significant differences between the total number of candidate studies obtained from each digital library, the most effective source seems to be ACM [5] with 18 out of 63 candidate studies.

Table 3.4: Search results for automated and manual searches.

Criteria	ACM	IEEEExplore	Science Direct	Springer Link	Google scholar	Sum
I1	+81	+50	+45	+1911	0	+2087
I2	0	0	0	0	+40	+40
E1	-51	-36	-32	-1882	0	-2001
E2	-11	-6	-6	-16	-23	-62
E3	-1	0	0	0	0	-1
Total	18	8	7	13	17	63

I1 - Included candidate studies from automatic search.

I2 - Included candidate studies from manual search.

E1 - Excluded studies that were not **SS** on software architecture.

E2 - Excluded duplicated studies.

E3 - Excluded studies that were not written in English.

3.3. STATE OF THE ART ON SOFTWARE ARCHITECTURE: AN EVIDENCE-BASED TERTIARY STUDY

Table 3.5: Selected studies for the tertiary study on software architecture.

Id	Paper
S1	Jamshidi, P., Ghafari, M., Ahmad, A., and Pahl, C. "A Framework for Classifying and Comparing Architecture-centric Software Evolution Research."Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on. IEEE, 2013.
S2	Weinreich, Rainer, and Iris Groher. "A fresh look at codification approaches for sakm: A systematic literature review."European Conference on Software Architecture. Springer, Cham, 2014.
S3	Me, Gianantonio, Coral Calero, and Patricia Lago. "A long way to quality-driven pattern-based architecting."European Conference on Software Architecture. Springer, Cham, 2016.
S4	Durelli, Rafael S., et al. "A mapping study on architecture-driven modernization."Information Reuse and Integration (IRI), 2014 IEEE 15th International Conference on. IEEE, 2014.
S5	Abdellatief, Majdi, et al. "A mapping study to investigate component-based software system metrics."Journal of Systems and Software 86.3 (2013): 587-603.
S6	Murugesupillai, Esan, Bardia Mohabbati, and Dragan Gašević. "A preliminary mapping study of approaches bridging software product lines and service-oriented architectures."Proceedings of the 15th International Software Product Line Conference, Volume 2. ACM, 2011.
S7	Aulkemeier, F., Schramm, M., Iacob, M. E., and Van Hillegersberg, J.. "A service-oriented e-commerce reference architecture."Journal of theoretical and applied electronic commerce research 11.1 (2016): 26-45.
S8	Javed, Muhammad Atif, and Uwe Zdun. "A systematic literature review of traceability approaches between software architecture and source code."Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering. ACM, 2014.
S9	Procaccianti, Giuseppe, Patricia Lago, and Stefano Bevini. "A systematic literature review on energy efficiency in cloud software architectures."Sustainable Computing: Informatics and Systems 7 (2015): 2-10.
S10	Rouhani, Babak Darvish, et al. "A systematic literature review on Enterprise Architecture Implementation Methodologies."Information and Software Technology 62 (2015): 1-20.
S11	Pourmirza, S., Peters, S., Dijkman, R., and Grefen, P.. "A systematic literature review on the architecture of business process management systems."Information Systems 66 (2017): 43-58.
S12	Guessi, M., Neto, V. V., Bianchi, T., Felizardo, K. R., Oquendo, F., and Nakagawa, E. Y.. "A systematic literature review on the description of software architectures for systems of systems."Proceedings of the 30th Annual ACM Symposium on Applied Computing. ACM, 2015.
S13	Alshuqayran, Nuha, Nour Ali, and Roger Evans. "A systematic mapping study in microservice architecture."Service-Oriented Computing and Applications (SOCA), 2016 IEEE 9th International Conference on. IEEE, 2016.
S14	Yang, Chen, Peng Liang, and Paris Avgeriou. "A systematic mapping study on the combination of software architecture and agile development."Journal of Systems and Software 111 (2016): 157-184.
S15	Breivold, Hongyu Pei, Ivica Crnkovic, and Magnus Larsson. "A systematic review of software architecture evolution research."Information and Software Technology 54.1 (2012): 16-40.
S16	Shahin, Mojtaba, Peng Liang, and Muhammad Ali Babar. "A systematic review of software architecture visualization techniques."Journal of Systems and Software 94 (2014): 161-185.
S17	Nikpay, F., Ahmad, R., Rouhani, B. D., and Shamshirband, S. . "A systematic review on post-implementation evaluation models of enterprise architecture artefacts."Information Systems Frontiers (2016): 1-20.

Continues on next page

Table 3.5 – Continuation from previous page

Id	Paper
S18	de Oliveira, Lucas Bueno Ruas. "A Systematic Review on Service-Oriented Reference Models and Service-Oriented Reference Architectures." Technical Report, São Paulo University. 2010.
S19	Martínez Fernández, Silverio Juan, et al. "Aggregating empirical evidence about the benefits and drawbacks of software reference architectures." Empirical Software Engineering and Measurement (ESEM), 2015 ACM/IEEE International Symposium on. 2015.
S20	Li, Zengyang, Peng Liang, and Paris Avgeriou. "Application of knowledge-based approaches in software architecture: A systematic mapping study." Information and Software Technology 55.5 (2013): 777-794.
S21	Tabares, Luis, Jhonatan Hernandez, and Ivan Cabezas. "Architectural approaches for implementing clinical decision support systems in cloud: a systematic review." Connected Health: Applications, Systems and Engineering Technologies (CHASE), 2016 IEEE First International Conference on. IEEE, 2016.
S22	Lytra, Ioanna, Stefan Sobernig, and Uwe Zdun. "Architectural decision making for service-based platform integration: A qualitative multi-method study." Software Architecture (WICSA) and European Conference on Software Architecture (ECSA), 2012 Joint Working IEEE/IFIP Conference on. IEEE, 2012.
S23	Guessi, M., Nakagawa, E. Y., Oquendo, F., and Maldonado, J. C. "Architectural description of embedded systems: a systematic review." Proceedings of the 3rd international ACM SIGSOFT symposium on Architecting Critical Systems. ACM, 2012.
S24	Ali, Nour, Sarah Beecham, and Ivan Mistrik. "Architectural knowledge management in global software development: a review." Global Software Engineering (ICGSE), 2010 5th IEEE International Conference on. IEEE, 2010.
S25	Lewis, Grace, and Patricia Lago. "Architectural tactics for cyber-foraging: Results of a systematic literature review." Journal of Systems and Software 107 (2015): 158-186.
S26	Salama, Maria, Rami Bahsoon, and Nelly Bencomo. "Managing trade-offs in self-adaptive software architectures: A systematic mapping study." Managing trade-offs in adaptable software architectures. 2017. 249-297.
S27	Williams, Byron J., and Jeffrey C. Carver. "Characterizing software architecture changes: A systematic review." Information and Software Technology 52.1 (2010): 31-51.
S28	Weyns, Danny, and Tanvir Ahmad. "Claims and evidence for architecture-based self-adaptation: a systematic literature review." European Conference on Software Architecture. Springer, Berlin, Heidelberg, 2013.
S29	Juziuk, Joanna, Danny Weyns, and Tom Holvoet. "Design patterns for multi-agent systems: a systematic literature Review." Agent-Oriented Software Engineering. Springer, Berlin, Heidelberg, 2014.
S30	Ameller, D., Burgués, X., Collell, O., Costal, D., Franch, X., and Papazoglou, M. P. "Development of service-oriented architectures using model-driven development: A mapping study." Information and Software Technology 62 (2015): 42-66.
S31	Maric, M., Matkovic, P., Tumbas, P., and Pavlicevic, V.. "Documenting Agile Architecture: Practices and Recommendations." EuroSymposium on Systems Analysis and Design. Springer, Cham, 2016.
S32	Dersten, Sara, Jakob Axelsson, and Joakim Froberg. "Effect analysis of the introduction of autosar: A systematic literature review." Software Engineering and Advanced Applications (SEAA), 2011 37th EUROMICRO Conference on. IEEE, 2011.
S33	Stelzer, Dirk. "Enterprise architecture principles: literature review and research directions." Service-oriented computing. ICSOC/ServiceWave 2009 workshops. Springer, Berlin, Heidelberg, 2010.

Continues on next page

3.3. STATE OF THE ART ON SOFTWARE ARCHITECTURE: AN EVIDENCE-BASED TERTIARY STUDY

Table 3.5 – Continuation from previous page

Id	Paper
S34	Herold, Sebastian, Martin Blom, and Jim Buckley. "Evidence in architecture degradation and consistency checking research: preliminary results from a literature review." <i>Proceedings of the 10th European Conference on Software Architecture Workshops</i> . ACM, 2016.
S35	Qureshi, Nadia, Muhammad Usman, and Naveed Ikram. "Evidence in software architecture, a systematic literature review." <i>Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering</i> . ACM, 2013.
S36	Gu, Qing, and Patricia Lago. "Exploring service-oriented system engineering challenges: a systematic literature review." <i>Service Oriented Computing and Applications</i> 3.3 (2009): 171-188.
S37	Zhu, Meng, Alf Inge Wang, and Hong Guo. "From 101 to nnn: a review and a classification of computer game architectures." <i>Multimedia systems</i> 19.3 (2013): 183-197.
S38	Tang, A., Razavian, M., Paech, B., and Hesse, T. M.. "Human aspects in software architecture decision making: a literature review." <i>Software Architecture (ICSA), 2017 IEEE International Conference on</i> . IEEE, 2017.
S39	Neto, C. R. L., Cardoso, M. P. S., Chavez, C. V. F. G., and de Almeida, E. S.. "Initial evidence for understanding the relationship between product line architecture and software architecture recovery." <i>Components, Architectures and Reuse Software (SBCARS), 2015 IX Brazilian Symposium on</i> . IEEE, 2015.
S40	Savolainen, Juha, and Varvana Myllarniemi. "Layered architecture revisited—Comparison of research and practice." <i>Software Architecture, 2009 & European Conference on Software Architecture. WICSA/ECSA 2009. Joint Working IEEE/IFIP Conference on</i> . IEEE, 2009.
S41	Haki, Mohammad Kazem, and Christine Legner. "New avenues for theoretical contributions in enterprise architecture principles-a literature review." <i>Trends in Enterprise Architecture Research and Practice-Driven Research on Enterprise Transformation</i> . Springer, Berlin, Heidelberg, 2012. 182-197.
S42	Saavedra, V., Dávila, A., Melendez, K., and Pessoa, M.. "Organizational maturity models architectures: a systematic literature review." <i>International Conference on Software Process Improvement</i> . Springer, Cham, 2016.
S43	Tofan, D., Galster, M., Avgeriou, P., and Schuitema, W.. "Past and future of software architectural decisions—A systematic mapping study." <i>Information and Software Technology</i> 56.8 (2014): 850-872.
S44	Affonso, F. J., Scannavino, K. R., Oliveira, L. B., and Nakagawa, E. Y.. "Reference architectures for self-managed software systems: a systematic literature review." <i>Software Components, Architectures and Reuse (SBCARS), 2014 Eighth Brazilian Symposium on</i> . IEEE, 2014.
S45	Arshad, Ali, and Muhammad Usman. "Security at software architecture level: A systematic mapping study." <i>Evaluation & Assessment in Software Engineering (EASE 2011), 15th Annual Conference on</i> . IET, 2011.
S46	Loya, S. R., Kawamoto, K., Chatwin, C., and Huser, V. "Service oriented architecture for clinical decision support: a systematic review and future directions." <i>Journal of medical systems</i> 38.12 (2014): 140.
S47	Molta, Y. H., Sarwar, A., Abbasi, M. A., and Jabeen, J.. "Software architecture and requirements: A systematic literature review." <i>Information and Communication Technologies (ICICT), 2015 International Conference on</i> . IEEE, 2015.
S48	Weinreich, Rainer, and Iris Groher. "Software architecture knowledge management approaches and their support for knowledge management activities: A systematic literature review." <i>Information and Software Technology</i> 80 (2016): 265-286.

Continues on next page

Table 3.5 – Continuation from previous page

Id	Paper
S49	Aleti, A., Buhnova, B., Grunske, L., Koziolk, A., and Meedeniya, I.. "Software architecture optimization methods: A systematic literature review."IEEE Transactions on Software Engineering 39.5 (2013): 658-683.
S50	Ahmad, Aakash, and Muhammad Ali Babar. "Software architectures for robotic systems: A systematic mapping study."Journal of Systems and Software 122 (2016): 16-39.
S51	Cruz-Benito, Juan, Roberto Therón, and Francisco J. García-Peñalvo. "Software architectures supporting human-computer interaction analysis: A literature review."International Conference on Learning and Collaboration Technologies. Springer, Cham, 2016.
S52	Stevanetic, Srdjan, and Uwe Zdun. "Software metrics for measuring the understandability of architectural structures: a systematic mapping study."Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering. ACM, 2015.
S53	Koziolk, Heiko. "Sustainability evaluation of software architectures: a systematic review."Proceedings of the joint ACM SIGSOFT conference–QoSA and ACM SIGSOFT symposium–ISARCS on Quality of software architectures–QoSA and architecting critical systems–ISARCS. ACM, 2011.
S54	Szvetits, Michael, and Uwe Zdun. "Systematic literature review of the objectives, techniques, kinds, and architectures of models at runtime."Software & Systems Modeling 15.1 (2016): 31-69.
S55	e Silva, Glauco de Sousa, et al. "Systematic Mapping of Architectures for Telemedicine Systems."International Conference on Computational Science and Its Applications. Springer, Berlin, Heidelberg, 2013.
S56	Palacios, Marcos, José García-Fanjul, and Javier Tuya. "Testing in Service Oriented Architectures with dynamic binding: A mapping study."Information and Software Technology 53.3 (2011): 171-189.
S57	Boucharas, V., van Steenbergen, M., Jansen, S., and Brinkkemper, S.. "The contribution of enterprise architecture to the achievement of organizational goals: a review of the evidence."International Workshop on Trends in Enterprise Architecture Research. Springer, Berlin, Heidelberg, 2010.
S58	Mahdavi-Hezavehi, Sara, Matthias Galster, and Paris Avgeriou. "Variability in quality attributes of service-based software systems: A systematic literature review."Information and Software Technology 55.2 (2013): 320-343.
S59	Groher, Iris, and Rainer Weinreich. "Variability support in architecture knowledge management approaches: a systematic literature review."System Sciences (HICSS), 2015 48th Hawaii International Conference on. IEEE, 2015.
S60	Ahmed, Musa Midila, and Sukumar Letchmunan. "A systematic literature review on challenges in service oriented software engineering."International Journal of Software Engineering and Its Applications 9.6 (2015): 173-186.
S61	Razavian, Maryam, and Patricia Lago. "A systematic literature review on SOA migration."Journal of Software: Evolution and Process 27.5 (2015): 337-372.
S62	Razavian, Maryam, and Patricia Lago. "A frame of reference for SOA migration."European Conference on a Service-Based Internet. Springer, Berlin, Heidelberg, 2010.
S63	Teka, Abdelneh Y., Nelly Condori-Fernandez, and Brahmananda Sapkota. "A systematic literature review on service description methods."International Working Conference on Requirements Engineering: Foundation for Software Quality. Springer, Berlin, Heidelberg, 2012.

Demography of the selected studies. Figure 3.5 shows the publication period (January 2009 to June 2017) of the systematic studies. From 2009 to 2016 there is a significant

3.3. STATE OF THE ART ON SOFTWARE ARCHITECTURE: AN EVIDENCE-BASED TERTIARY STUDY

increase in the number of systematic studies. We cannot state the exact reason for this difference in the number of articles published per year, however, the number of articles in 2016 has grown abnormally (more than five times greater than the number of publications in 2009). In general terms, the result show the increasing interest in the systematic studies in the software architecture research since 2009.

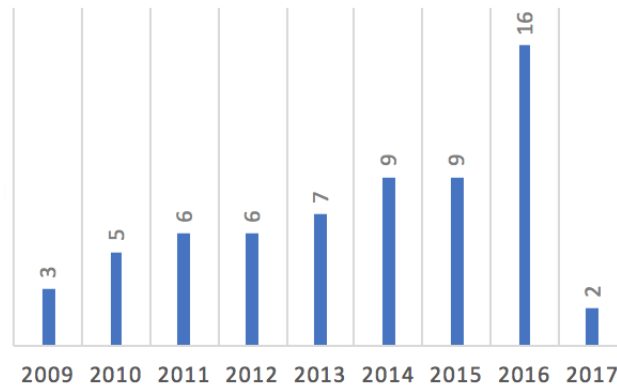


Figure 3.5: Publication period of the selected systematic studies.

Quality assessment of Systematic Studies. We assessed the quality of each selected study as per the Quality Assessment (QA) criteria discussed previously. The total scores of each systematic study are listed in the Table 3.6. As the scale used is from 0 to 4, the score 2 is considered the neutral. In other words, a selected study with QA score > 2 is considered to have good quality. We identified 30 studies with QA score > 2 (50.84%), 17 studies with QA score $= 2$ (28.81%), and 16 studies with QA score < 2 (27.11%). More details are given in the next section.

Table 3.6: List of selected systematic studies.

ID	Year	Venue	Publisher	Review Type	QA Score
S1	2013	CSMR	IEEE	SLR	3.5
S2	2014	ECSA	Springer	SLR	4
S3	2016	ECSA	Springer	SMS	2
S4	2014	IRI	IEEE	SMS	2.5
S5	2012	JSS	Elsevier	SMS	3.5
S6	2011	SPLC	ACM	SMS	2.5
S7	2016	JTAER	Universidad de Talca	SLR	3
S8	2014	EASE	ACM	SLR	2.5
S9	2014	SUSCOM	Elsevier	SLR	1.5
S10	2016	IST	Elsevier	SLR	2.5
S11	2017	Information Systems Journal	Elsevier	SLR	2.5
S12	2015	SAC	ACM	SLR	2
S13	2016	SOCA	IEEE	SMS	1.5
S14	2016	JSS	Elsevier	SLR	2

Continues on next page

Table 3.6 – Continuation from previous page

ID	Year	Venue	Publisher	Review Type	QA Score
S15	2011	IST	Elsevier	SLR	3.5
S16	2014	JSS	Elsevier	SLR	3.5
S17	2016	Information Systems Fron- tiers	Springer	SLR	3
S18	2010	ECSA	Springer	SLR	2
S19	2015	ESEM	IEEE	SLR	3
S20	2013	IST	Elsevier	SMS	3.5
S21	2016	CHASE	IEEE	SLR	3
S22	2012	WICSA-ECSA	IEEE	SLR	2.5
S23	2012	ISARCS	ACM	SLR	2
S24	2010	ICGSE	IEEE	SLR	2
S25	2015	JSS	Elsevier	SLR	1.5
S26	2016	Managing Trade-Offs in Adaptable Software Archi- tectures	Elsevier	SMS	2.5
S27	2010	IST	Elsevier	SLR	3
S28	2013	ECSA	Springer	SLR	2.5
S29	2014	Agent-Oriented Software En- gineering	Springer	SLR	2.5
S30	2015	IST	Elsevier	SMS	2
S31	2016	SIGSAND / PLAIS	Springer	SLR	0.5
S32	2011	SEAA	IEEE	SLR	2
S33	2009	ICSOC	Springer	SLR	2
S34	2016	ECSAW	ACM	SLR	1.5
S35	2013	EASE	ACM	SLR	2.5
S36	2009	SOCA	Springer	SLR	2
S37	2013	Multimedia Systems	Springer	SLR	1.5
S38	2017	ICSA	IEEE	SLR	2
S39	2015	SBCARS	IEEE	SLR	2.5
S40	2009	WICSA-ECSA	IEEE	SLR	1
S41	2012	TEAR	Springer	SLR	1.5
S42	2016	CIMPS	Springer	SLR	3.5
S43	2014	IST	Elsevier	SMS	3
S44	2014	SBCARS	IEEE	SLR	2
S45	2011	EASE	IET	SMS	1.5
S46	2014	Journal of Medical Systems	Springer	SLR	2
S47	2016	ICICT	IEEE	SLR	1
S48	2016	IST	Elsevier	SLR	3.5
S49	2012	TSE	IEEE	SLR	2
S50	2016	JSS	Elsevier	SLR	2
S51	2016	LCT	Springer	SLR	1.5
S52	2015	EASE	ACM	SLR	2
S53	2011	QoSA-ISARCS	ACM	SLR	3.5
S54	2016	SoSyM	Springer	SLR	2
S55	2013	ICCSA	Springer	SMS	1.5
S56	2011	IST	Elsevier	SMS	3.5
S57	2010	TEAR	Springer	SLR	3

Continues on next page

3.3. STATE OF THE ART ON SOFTWARE ARCHITECTURE: AN EVIDENCE-BASED TERTIARY STUDY

Table 3.6 – Continuation from previous page

ID	Year	Venue	Publisher	Review Type	QA Score
S58	2013	IST	Elsevier	SLR	3.5
S59	2015	HICSS	IEEE	SLR	3.5
S60	2015	IJSEIA	SERSC	SLR	2
S61	2015	Journal of Software: Evolution and Process	Wiley	SLR	2
S62	2010	European Conference on a Service-Based Internet	Springer	SLR	2
S63	2012	REFSQ	Springer	SLR	3

3.3.3 Reporting: Answering the research questions

Based on the analysis of the extracted data from the selected studies, the following paragraphs discuss the findings that help answering each research question.

RQ1.1: How many primary studies are included in these reviews? The 63 secondary studies selected analyze a total of 3092 primary studies. The “primary study count” row of Table 3.7 shows the number of primary studies of each review. [S35] is the secondary study with more primary studies (230 studies) and [S19] is the one with less (5 in total). 8 out of the 63 secondary studies (12.7%) have more than 100 selected primary studies [S28, S30, S34, S35, S43, S48, S49, S54], 12 of the secondary studies (19%) have between 50 and 99 [S1, S3, S14, S20, S25, S36, S38, S42, S50, S60], 41 of the secondary studies (65.1%) have less than 50 selected primary studies [S2, S4-S13, S17-S19, S21, S23, S24, S26, S27, S29, S31-S33, S37, S39-S41, S44-S47, S51, S52, S55-S59, S61-S63], and 2 secondary studies (3.2%) do not describe the number of primary studies selected [S22, S53].

RQ1.2: What is the time span covered by these reviews? A good practice is to inform about the time span (start and end dates) of the search. This is shown in Table 3.7 (see “Time span” row). 21 out of 63 secondary studies (33.3%) do not use a start date for searches of primary studies. So, these reviews include all published works for their specific topics [S4, S12, S15, S18, S19, S22-S26, S31, S32, S34, S40, S42, S44, S46, S47, S53, S54, S57], widening their search space. S35 is the secondary study that defines the oldest start time (1972). Regarding the time in which each review was performed, the most recent one is from 2016 [S13, S21, S31, S34] and the oldest is from 2001 [S16].

RQ1.3: Is the list of the included primary studies available? The row “Primary study list” in Table 3.7 informs if the secondary study lists the selected primary studies. 14 out of 63 secondary studies (22.2%) do not include the list of primary studies [S3, S4, S12, S22, S28, S31, S32, S34-S36, S46, S53, S60, S62]. The reason may be due to the limited number of pages imposed by the publishers. This is most unfortunate, as it does not facilitate replication nor traceability.

Table 3.7: Number of primary studies per review, time span of review and if the review shows the primary study list

#	Primary study count	Time span	Primary study list	#	Primary study count	Time span	Primary study list
S1	60	1995-2011	yes	S33	11	1990-2007	yes
S2	7	2008-2013	yes	S34	119	Until 2016	no
S3	99	1990-2015	no	S35	230	1972-2013	no
S4	30	Until 2013	no	S36	51	2000-2008	no
S5	31	2000-2010	yes	S37	40	1990-2009	yes
S6	48	2002-2010	yes	S38	81	2005-2015	yes
S7	48	2003-2013	yes	S39	28	1998-2015	yes
S8	11	1999-2013	yes	S40	11	Until 2008	yes
S9	26	2000-2013	yes	S41	19	1990-2011	yes
S10	46	1997-2013	yes	S42	70	Until 2015	yes
S11	41	1994-2015	yes	S43	144	2002-2012	yes
S12	38	Until 2013	no	S44	22	Until 2013	yes
S13	33	2014-2016	yes	S45	40	1998-2011	yes
S14	54	2001-2014	yes	S46	44	Until 2013	yes
S15	82	Until 2010	yes	S47	29	Until 2013	yes
S16	53	1999-2001	yes	S48	115	2004-2015	yes
S17	34	2006-2014	yes	S49	188	1992-2011	yes
S18	21	Until 2009	yes	S50	56	1991-2015	yes
S19	5	Until 2014	yes	S51	16	1998-2013	yes
S20	55	2000-2011	yes	S52	25	1990-2013	yes
S21	22	2010-2016	yes	S53	-	Until 2010	no
S22	-	Until 2012	no	S54	122	Until 2013	yes
S23	24	Until 2011	yes	S55	37	1996-2011	yes
S24	25	Until 2009	yes	S56	37	2000-2009	yes
S25	58	Until 2013	yes	S57	14	Until 2010	yes
S26	20	Until 2015	yes	S58	46	2000-2011	yes
S27	23	1997-2007	yes	S59	13	2008- 2013	yes
S28	102	2000-2012	no	S60	91	2009-2014	no
S29	39	1998-2012	yes	S61	31	2000-2013	yes
S30	129	2003-2013	yes	S62	44	2000-2010	no
S31	10	Until 2016	no	S63	24	2002-2010	yes
S32	20	Until 2010	no				

RQ1.4: Is the quality of the primary studies assessed? We have analyzed the limitations of the existing systematic studies based on the quality assurance score (see Section 3.3.1). Regarding QA1 (inclusion and exclusion criteria for selecting primary studies), the results show that the quality of the studies is very good. 52 out of the 63 secondary studies (82.5%) explicitly describe the inclusion and exclusion criteria (QA score = 1) [S1-S30, S32, S35, S36, S39, S42-S44, S46, S48-S50, S52-S56, S58-S63], 10 studies (15.9%) show the implicit inclusion and exclusion criteria (QA score = 0.5) [S33, S34, S37, S38, S40, S41, S45, S47, S51, S57], and only one study (1.6%) does not describe any of these criteria [S31].

Regarding QA2, we found that 17 secondary studies (27%) explicitly describe using manual and automatic search on four or more digital libraries [S1, S2, S5, S6, S16, S20,

S26, S29, S33, S39, S42, S43, S48, S56, S57-S59], 43 secondary studies (68.2%) show the implicit manual and automatic search or do not use more than three digital libraries [S3, S4, S7, S10-S12, S14, S15, S17-S19, S21-S25, S27, S28, S30-S32, S34-S38, S40, S41, S41-S47, S49-S55, S60-S63], and only three secondary studies (4.8%) do not describe any use of manual or automatic search [S8, S9, S13].

With respect to QA3, we found that 23 secondary studies (38.3%) scored full mark (score = 1) as all 63 secondary studies reported on the evaluation of the quality assessment of the primary studies [S1, S2, S5, S7, S8, S10, S15-S17, S19-S22, S27, S28, S42, S48, S53, S56-S59, S63]. In one secondary study the QA score is 0.5 [S35]. Surprisingly, in 39 secondary studies, the quality assessment of primary studies (62%) is not considered [S3, S4, S6, S9, S11-S14, S18, S23-S26, S29-S34, S36-S41, S43-S47, S49-S52, S54, S55, S60-S62]. This is an important limitation that the researchers should address in future systematic studies.

Regarding QA4, only 7 secondary studies (11.1%) give details about the primary studies individually [S2, S4, S11, S15, S38, S43, S53]. In 48 secondary studies (76.2%) show the information about the primary studies in an aggregated form [S1, S3, S5-S9, S12-S14, S16-S21, S23, S24, S26, S27, S29, S30, S32-S37, S39, S41, S42, S44-S46, S48-S52, S54, S56-S63] and 8 secondary studies (12.7%) do not describe the primary studies data adequately [S10, S22, S25, S28, S31, S40, S47, S55].

RQ1.5: Who is the secondary study targeted to? 6 secondary studies (9.5%) are targeted to practitioners [S3, S25, S29, S32, S47, S58], 20 secondary studies (31.7%) to researchers [S4, S6, S26, S30, S34, S36, S38, S40, S41, S44, S45, S49, S51, S53, S54, S56, S59, S61-S63], and 26 secondary studies (41.3%) to both practitioners and researchers [S1, S5, S7, S8, S10, S13-S21, S28, S33, S35, S37, S39, S43, S46, S48, S50, S52, S57, S60]. 11 secondary studies (17.5%) do not describe their target [S2, S9, S11, S12, S22-S24, S27, S31, S42, S55].

RQ1.6: Which publication venues are most commonly used? The goal is to find out the publication types and the publication venues where the selected systematic studies are published. 34 secondary studies (53.9%) are published in conferences [S1-S4, S6, S8, S12, S13, S18, S19, S21-S24, S28, S31, S32, S34-S36, S38-S40, S42, S44, S45, S47, S51-S53, S55, S59, S62, S63], followed by 25 secondary studies (39.7%) in journals [S5, S7, S9-S11, S14-S17, S20, S25, S27, S30, S37, S41, S43, S46, S48-S50, S54, S56, S58, S60, S61], 2 secondary studies (3.1%) in workshops [S33, S57], and 2 secondary studies (3.1%) in others venues (e.g., Universities publication) [S26, S29].

Regarding where the selected systematic studies were published, we found that Information and Software Technology journal (IST has 9 studies — 14.2%) as the most targeted venue [S10, S15, S20, S27, S30, S43, S48, S56, S58]. Next, Journal of Systems and Software (JSS has 5 secondary studies — 7.9%) [S5, S14, S16, S25, S50], European Conference on Software Architecture (ECSA has 4 secondary studies — 6.3%) [S2, S3, S18, S28],

international conference on Evaluation and Assessment in Software Engineering (EASE has 4 secondary studies — 6.3%) [S8, S35, S45, S52], Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture (WICSA-ECSA has 2 secondary studies — 3.1%) [S22, S40], IEEE International Conference on Service-Oriented Computing and Applications (SOCA has 2 secondary studies — 3.1%) [S13, S36], Brazilian symposium on components, architecture and software reuse (SBCARS has 2 secondary studies — 3.1%) [S39, S44], and others 35 different venues with 1.6% each [S1, S4, S6, S7, S9, S11, S12, S17, S19, S21, S23, S24, S26, S29, S31-S34, S37, S38, S41, S42, S46, S47, S49, S51, S53-S55, S57, S59-S63].

RQ2: What is the current status of consolidation of data collected from different literature reviews on software architecture? The answer to this question started with an analysis of both, the topics of interest of WICSA and the JSS special issue on new frontiers in software architecture. Each of these topics were associated to a category against which the research questions of the secondary studies were mapped. It is important to notice that some secondary studies were mapped in more than one category (e.g., S6 was mapped to CAT25 and CAT1). Table 3.8 shows the result of this mapping process, together with the identifier of secondary study and the total number of studies in each category. The category *Developments in architectural design, analysis and evaluation* (CAT1) emerged as the most researched by the 63 secondary studies.

The findings for the categories are summarized next, following their order of appearance in Table 3.8, except for CAT9-CAT13, CAT15, CAT16, CAT21, CAT23, and CAT24 which have no secondary studies associated.

– **CAT1: Developments in architectural design, analysis and evaluation.** The studies from CAT1 include a wide range of topics, such as metrics definition and methods for architectural evaluation [S5, S6, S17, S18, S27, S52, S53, S56], quality attributes analysis [S3, S30, S43, S45, S52, S53, S58, S60, S63], design strategies for specific types of development environments (e.g., cyber-foraging [S25], model-driven development [S30], automotive open software architecture [S32], component-based architecture [S5, S51], service-based architecture [S58, S7, S13, S36, S46, S63], decision support for clinical systems [S46, S21], computer game architecture [S37], benefits and best practices of use of reference architectures [S7, S19, S22, S23], use of architectural patterns [S22, S29], architecture erosion [S35], architecture for web application [S47], and human aspects in software architecture decision making [S51, S38].

– **CAT2: Management of architectural knowledge, decisions, and rationale.** The research questions of the secondary studies concerning CAT2 point to the importance of documenting architectural decisions to mitigate the “architectural vaporization knowledge” during architecture evolution [S59, S20, S24] and also highlight that there are still

3.3. STATE OF THE ART ON SOFTWARE ARCHITECTURE: AN EVIDENCE-BASED TERTIARY STUDY

Table 3.8: Categories and corresponding number of secondary studies

Id	Category	Studies	Quantity
CAT1	Developments in architectural design, analysis and evaluation	S3, S5-S7, S13, S17-S19, S21-S23, S25, S29, S30, S32, S35-S38, S40, S43, S45-S47, S51-S53, S56, S58, S60, S63	31
CAT2	Management of architectural knowledge, decisions, and rationale	S2, S20, S24, S48, S59	5
CAT3	Innovative architecture centric process, models and frameworks	S8	1
CAT4	Software architecture and agility	S14, S31	2
CAT5	Architecture-centric model driven engineering	S30, S54	2
CAT6	Architectures for reconfigurable and self-adaptive systems	S26, S28, S44	3
CAT7	Architectures for ultra-large scale, long-lived systems and systems-of-systems	S12	1
CAT8	Software architecture and the cloud	S9, S21	2
CAT9	Software architecture and virtualization		0
CAT10	Software architecture and big data		0
CAT11	Architectures for cyber-physical systems		0
CAT12	Architectural concerns of autonomic systems		0
CAT13	Software architecture and system architecture, including software-defined networking		0
CAT14	Software tools and environments for architecture-centric software engineering	S1, S8, S10, S14	4
CAT15	Industrial applications, case studies, best practices and experience reports		0
CAT16	Architecting the internet of things		0
CAT17	Architectural reconstruction techniques, refactoring and evolving architecture design decisions and solutions	S1, S4, S15, S27, S34, S39, S49, S61, S62	9
CAT18	Architecture description languages	S12, S16	2
CAT19	Energy-awareness and sustainability	S9, S53	2
CAT20	Software architectures for emerging systems	S25, S37, S50, S55	4
CAT21	Software architecture for legacy systems and systems integration		0
CAT22	Cultural, economic, business and managerial aspects of software architecture	S10, S11, S17, S33, S41, S42, S57	7
CAT23	Software architects' roles and responsibilities		0
CAT24	Training, education, and certification of software architects		0
CAT25	Open architectures, product-line architectures, software ecosystems	S6, S39, S58, S59	4

many critical open issues that need to be addressed to enable widespread adoption of architectural Knowledge maintenance in industry (e.g., cost-efficient capturing, long-term perspective, holistic approaches, and industrial validation) [S2, S48].

– **CAT3: Innovative architecture centric process, models and frameworks.** The study associated with CAT3 focus on traceability approaches between software architecture and

source code [S8]. It highlights that most of the existing traceability approaches between software architecture and source code provide insufficient support to address the various architectural concerns. For instance, there is a need to more precisely address various stakeholder issues, quality concerns, and software development artifacts across various architectural or design views.

– **CAT4: Software architecture and agility.** Regarding CAT4, only [S14] and [S31] report on agile development. [S14] verifies which architectural activities can be practiced in an agile context and what are the costs and benefits, challenges and factors that have an impact on the success of applying the architecture-agility combination. [S31] focuses on the documentation of the agile architecture and it highlights that agile practitioners do not renounce architecture documenting activities, but rather consider them significant in the development of complex systems.

– **CAT5: Architecture-centric model driven engineering.** For CAT5, [S30] shows what are the characteristics of model-driven development supporting service-oriented architectures (SOA), which SOA is supported, and how these approaches handle non-functional requirements. [S54] identifies different objectives of architectures of models at runtime (e.g., adaptation, abstraction, consistency, conformance, error handling, monitoring, simulation, prediction, platform independence, and policy checking/enforcement). It also identifies different techniques when processing runtime models to extract runtime data, raise the abstraction level and enforce constraints (e.g., introspection, model conformance, model comparison, model transformation, and model execution).

– **CAT6: Architectures for reconfigurable and self-adaptive systems.** Studies addressing CAT6 show how reference architectures have been designed for self-adaptive systems, as well as what are the claims made and which techniques and kinds of models have been used [S26, S28, S44].

– **CAT7: Architectures for ultra-large scale, long-lived systems and systems-of-systems.** CAT7 is addressed in [S12], focusing on the how system-of-systems software architectures have been described and highlights the lack of consensus on how to better deal with these descriptions.

– **CAT8: Software architecture and the cloud.** For CAT8, we found [S9] and [S21] focusing on architecture for cloud system. [S9] gives a state-of-the-art of how energy efficiency is addressed by cloud software architectures while [S21] focuses on architectural approaches for implementing Clinical Decision Support Systems (CDSS) in the cloud.

– **CAT14: Software tools and environments for architecture-centric software engineering.** For studies addressing CAT14, we noticed the authors' interest in synthesising

the existing data about tool support for enterprise architecture implementation [S10], architecture agility [S14], architecture evolution [S1], and traceability between software architecture and the source code [S8].

– **CAT17: Architectural reconstruction techniques, re-factoring and evolving architecture design decisions and solutions.** For CAT17, the studies show that tools can be useful to facilitate the enterprise architecture development [S34] and to manage the impact of a change in a software architecture [S27]. However, most of the existing tools are proof of concept [S1], and there is a lack of supporting tools for refactoring [S4], product line architecture [S39], SOA migration [S61, S62], automatic creation of architectural models [S49], and orchestration of the evolution of different platforms, decision nodes, organizations and processes [S15].

– **CAT18: Architecture description languages.** For CAT18, we found two secondary studies focusing on architecture description [S12, S16]. [S12] identifies that more research is needed for effectively using architecture descriptions in the evaluation and evolution of system-of-systems and [S16] concludes that little attention has been paid to carry out controlled experiments to evaluate and compare the usefulness (e.g., ease of use, effectiveness, efficiency, application cost) of the reported visualization techniques and tools due to the excessive effort and resources needed.

– **CAT19: Energy-awareness and sustainability.** For CAT19, [S9] highlights that Self-Adaptation is the most adopted strategy to achieve energy efficiency and that cloud federation will need more research in the future, due to the diffusion of multi-cloud environments and the need of optimizing the usage of Cloud infrastructures. In addition, [S53] highlights that scenario-based methods should better validate their potential return on investment.

– **CAT20: Software architectures for emerging systems.** For CAT20, the studies show that there are gaps and opportunities for research in quality attributes that are relevant to cyber-foraging systems (e.g., ease of distribution and installation, resiliency, and security) [S25], that multi-server architectures (e.g., nnn Graph Style) and Peer-to-Peer architectures (e.g., n0n Graph Style) are the two major architectural patterns for MMOG in game architecture research [S37], that there is a growth of research with various innovative solutions for robotics software architecture (e.g., model-driven and cloud-based robotics solutions) [S50], and that the client-server architectures are the most used architectural style to build telemedical applications [S55].

– **CAT22: Cultural, economic, business and managerial aspects of software architecture.** For CAT22, the studies focus on three different topics: enterprise architecture

[S33, S41, S10, S17, S57], business process management [S11], and organizational maturity model [S42]. Regarding the enterprise architecture, the studies describe that there are various gaps in enterprise architecture definition (e.g., there is no accepted definition of enterprise architecture principles) [S33, S41], practices that need to be considered to provide a useful enterprise architecture implementation (e.g., architectural design and requirement management) [S10], lack of methodologies or frameworks for enabling enterprise architecture evaluation [S17], and most existing research on enterprise architecture is targeting information technology and information technology-related effects of enterprise architecture [S57]. Regarding the business process management topic, [S11] verifies that although extensive research has been carried out on the architectures of business process management systems for a particular company, only a small part of them have appropriately covered inter-organizational collaboration as well. Regarding the organizational maturity model topic, [S42] identifies and compares nine architectural styles (e.g., SW-CMM-based model) of organizational maturity models for different domains.

– **CAT25: Open architectures, product-line architectures, software ecosystems.** For CAT25, the studies show that the main motivating factor for the research papers in software product lines was variability management [S6, S58], but few existing works focus on fully automated recovery of variability at the architectural level [S39]. In addition, [S59] describes that a broad application of software architecture knowledge management in the context of variability and software product lines is still missing.

RQ2.2: Is there some secondary study focusing on methods for the derivation of software architecture models from requirements specifications? We found 31 systematic reviews and mapping studies on software architecture in the literature, focusing on developments in architectural design, analysis and evaluation (CAT1). However, even though several works discuss requirements at the architectural level, they do not address our research questions.

RQ3.1: How many systematic reviews, including systematic literature reviews and mapping studies, addressing any topic in software architecture, are available? SLRs is the most frequently used systematic method in the software architecture domain with 51 studies (80.9%) [S1, S2, S7-S12, S14-S19, S21-S25, S27-S29, S31-S42, S44, S46-S54, S57-S63], almost 4 times more studies than SMS with 12 studies (19.1%) [S3-S6, S13, S20, S26, S30, S43, S45, S55, S56]. Kitchenham *et al.* [152] describe a pragmatic comparison between SLRs and SMSs and, according to this comparison, we would say that all selected systematic studies are, in fact, SMSs given that they classify and perform a thematic analysis of specific topics on software architecture, with more generic research questions and with a broader scope.

RQ3.2: What is the origin of these reviews? Regarding the countries, most of the studies (47 out of 63) were co-authored by researchers from European countries [S1-S4, S7-S9, S11-S16, S19, S20, S22-S26, S28, S30-S34, S36-S38, S40, S41, S43, S48-S59, S61-S63] and researchers from South American [S4, S12, S18, S19, S21, S23, S39, S42, S44, S55] and Asian [S5, S10, S14, S16, S17, S20, S35, S45, S47, S60] countries contributed with 10 studies each. Next, the Oceania countries contributing with 4 studies [S16, S38, S43, S49] and North American countries contributed with 3 systematic studies as we observed from collected data [S6, S27, S46].

Table 3.9 shows the number of reviews and authors per country. This shows that The Netherlands and Brazil are the countries with more authors performing systematic studies on software architecture (with 27 and 26 authors respectively) and as a consequence they also are the countries with more selected studies (with 14 and 9 respectively).

Table 3.9: Number of reviews and authors per country





























	Country	Total authors	Total studies
	The Netherlands	27	14
	Brazil	26	9
	Spain	16	5
	Sweden	11	5
	Malaysia	10	3
	Austria	8	7
	Pakistan	8	3
	United Kingdom	8	3
	Germany	6	5
	Ireland	6	3
	Australia	4	3
	France	4	3
	Serbia	4	1
	USA	4	2
	China	3	3
	Colombia	3	1
	Italy	3	2
	Norway	3	1
	Perú	3	1
	Switzerland	3	2
	Finland	2	1
	Iran	2	2
	Belgium	1	1
	Canada	1	1
	Czech Republic	1	1
	Denmark	2	1
	New Zealand	1	1
	Portugal	1	1

Table 3.10 shows the author's affiliation with more than two reviews on software architecture. Vrije Universiteit Amsterdam (The Netherlands) is the institution with more reviews published, seven in total, followed by Universidade de São Paulo (Brazil) with 6 in total.

Table 3.10: Number of reviews per author's affiliation with at least 2 studies.

Affiliation	Reviews	Total
Vrije Universiteit Amsterdam	S03, S09, S16, S25, S36, S61, S62	7
Universidade de São Paulo	S04, S12, S18, S23, S42, S44	6
University of Groningen	S14, S20, S43, S58	4
University of Vienna	S08, S22, S52, S54	4
Johannes Kepler University Linz	S02, S48, S59	3
Wuhan University	S14, S16, S20	3
ABB Corporate Research	S15, S53	2
International Islamic University	S35, S45	2
Linnaeus University	S28, S29	2
Mälardalen University	S15, S32	2
Swinburne University of Technology	S38, S49	2
Universitat Politècnica de Catalunya	S19, S30	2
University of Limerick	S24, S34	2
University of Malaya	S10, S17	2
University of South Brittany	S12, S23	2
University of Twente (NL)	S07, S63	2

RQ3.3: What is the impact of these reviews, in terms of citations? Figure 3.6 shows the number of citation of each review. The most cited review is [S49] with 186 citations, followed by [S15] with 123 citations and by [S27] with 115 citations. The other studies have less than 100 citations. The selected review's set h-index⁴ is 22. In this case, we use it to assess the impact of the body of publications included in this tertiary study.

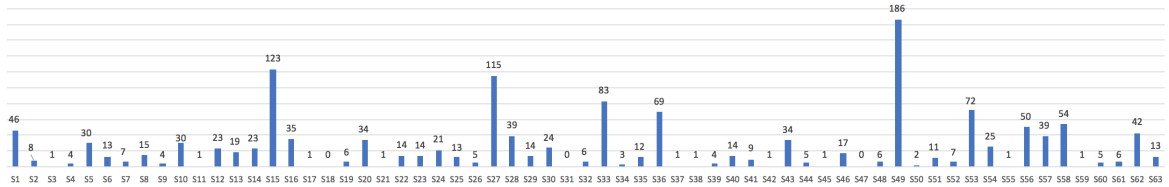


Figure 3.6: Number of citations per secondary study.

3.3.4 Threats to validity

The validity threats of this study are related to potential problems associated to the completeness of our search query and research sources, the secondary studies selection process, as well as the care and rigorous execution by the authors of the secondary studies. This is a serious issue because if the quality of the included primary studies is not evaluated, threats to the validity of the results published by those secondary studies may put at risk the goodness of the results. To mitigate this risk, we only considered papers published in peer-reviewed venues as this suggests that the studies were evaluated by more than one specialist in the area, decreasing the risk of accepting low quality studies. Also, we considered the use of both automatic and manual search in more than four

⁴The h-index is the largest number h such that h publications have at least h citations [118]. The h-index is often used to characterize the scientific output of a researcher.

different digital libraries.

Before and during the execution process of our tertiary study, all steps (e.g., the definition of the research questions, identification of secondary studies, identification of bibliographic sources, selection of secondary studies, and results) were independently validated by external reviewers (including faculty members and [Ph.D.](#) students who are members of NOVA LINCS research laboratory) as recommended in [\[149\]](#). Even though our research string includes the relevant keywords and we performed an external validation, there is always a small risk that some secondary studies were not included in our list of selected papers because either they are not indexed by the source libraries used, or they do not include the relevant keywords in their title and abstract. To mitigate this risk, we performed a manual search in Google Scholar [\[102\]](#).

Regarding the limitations of quality assessment, we have followed the QA criteria suggested by Kitchenham *et al.* [\[150\]](#) which is more suitable for systematic studies. However, we believe that a lower score for studies does not mean that the quality of the results is bad. Instead, we just believe that studies do not follow all the good practices suggested by the [EBSE](#) community because there is no definition of what a high level quality study is [\[149\]](#).

3.4 Final considerations

This Chapter offered an overview of software architecture, service-oriented architecture, model-driven architecture, and, finally, aggregated consolidated findings on software architecture through a tertiary study. From this study, we emphasize next some findings regarding each of the three main questions.

RQ1: What is the currently available information concerning software architecture, systematically aggregated by means of a systematic literature reviews, or a mapping study? The 63 selected systematics studies collect 3092 primary studies (RQ1.1) from 1972 until 2016 (RQ1.2). 77.8% of these studies ensure the traceability of their findings with the primary studies, providing the list of these primary studies (RQ1.3). However, 62% of the selected reviews do not assess the quality of the studies they analyzed (RQ1.4), showing an important limitation that researchers must address in future systematic studies on software architecture. Most of reviews (41.3%) are targeted to both researchers and practitioners (RQ1.5). In addition, despite the fact that most of the studies are published in conferences (53.9%), the Information and Software Technology journal is the largest publisher (14.2%) of systematic studies on software architecture (RQ1.6).

RQ2: What is the current status of consolidation of data collected from different literature reviews on software architecture? Based on our classification scheme with 25 categories, we identified the topics of research with more secondary studies, and the list of topics not yet covered by such type of studies. Regarding the topics of research with

systematic studies, we verified that *CAT 1 – developments in architectural design, analysis and evaluation* – is the category (topic of research) with more secondary study (31 out of 63). In the future, we plan to update this tertiary study and we identified a need to create a sub-classification of the secondary studies into this CAT1 to clarify the real distribution of studies on architectural design, analysis, and evaluation. We believe that there are sub-topics into CAT1 which have not yet been addressed by systematic studies. For example, no secondary studies looking for software architecture derivation methods from requirements specifications were found. Given that this topic is essential to answer our [Ph.D.](#) research question, we carried out the systematic mapping described in [Chapter 4](#). We found that 40% of the categories (10 out of 25 categories) have no secondary studies associated. Some of these categories are software architecture and virtualization, software architecture for big data, cyber-physical systems, networking systems, autonomic systems, and internet of things, empirical approaches to evaluate software architectures, software architects' roles and responsibilities, and training, education, and certification of software architects.

RQ3: Who is performing literature surveys? Systematic literature reviews (SLR) is the most frequently used systematic method (RQ3.1). However, according to the comparison between [SLR](#) and [SMS](#) [152], we believe that the selected [SLR](#) are, in fact, [SMSs](#) given that those studies classified and performed a thematic analysis of specific topics on software architecture, with somewhat generic research questions and with a broader scope. Most of the selected secondary studies were co-authored by researchers from Europe where The Netherlands is the country with more authors and secondary studies (RQ3.2) and the second is Brazil in South American (RQ3.2). Vrije Universiteit Amsterdam (The Netherlands) is the institution with more secondary studies published, seven in total, followed by Universidade de São Paulo (Brazil) with 6 in total (RQ3.2). Finally, the set of 63 secondary studies selected have an h-index of 22, showing that the impact of these studies is significant.

DERIVING ARCHITECTURAL MODELS FROM REQUIREMENTS SPECIFICATIONS

Software architecture derivation methods offer ways to derive architectural models from requirements specifications. These models must balance different forces that should be analyzed during this derivation process, such as those imposed by different application domains and quality attributes. Such balance is difficult to achieve, requiring skilled and experienced architects. Our tertiary study indicates the nonexistence of secondary studies looking for software architecture derivation methods. Hence, the purpose of this Chapter is to perform a second systematic mapping study to provide a comprehensive overview of the existing methods to derive architectural models from requirements specifications and offer a research roadmap to challenge the community to address the identified limitations and open issues that require further investigation. This study follows the traditional *planning*, *conducting*, and *reporting* phases. This study resulted in 39 primary studies selected for analysis and data extraction, from the 2575 documents initially retrieved. The major findings indicate that current architectural derivation methods rely heavily on the architects' tacit knowledge (experience and intuition), do not offer sufficient support for inexperienced architects, and lack explicit evaluation mechanisms. These and other findings are synthesized in a research roadmap which results would benefit researchers and practitioners.

4.1 Planning

To plan a mapping study, we start by formulating the research questions and the search string to run in the digital libraries, next we define the search strategies and the research sources and studies, then we determine how to assess the quality of the studies and specify what data should be extracted from the selected studies, and finally, we review

the protocol, seeking external reviewers to validate and offer any additional suggestions for improvement.

4.1.1 Formulating the research questions

We used the **PICOC** method [205], which is recommended to simplify the construction of the research question. It offers a multi-view definition of the research objective, highlighting (in bold in the table) the main research terms.

Table 4.1: PICOC Analysis

Population	Papers focusing on the transition from Requirements to Software Architecture , considering all types of industries, systems and application domains.
Intervention	Software architecture derivation methods that use any methodology, tool, technology, and procedures.
Comparison	Not applicable: our intention is to classify the existing derivation methods based on specific criteria nor to compare the methods with other methods or processes.
Outcome	Overview on the context, benefits to the users, content, and validation of the software architecture derivation methods.
Context	Research papers. We are working in a research context with experts in the domain as well as other practitioners, academics, consultants and students.

This lead to the definition of the following research question:

RQ: *What methods have been proposed for the derivation of software architecture models from requirements specifications?*

4.1.2 Formulating the search string

The key terms in our research question led to the definition of the search query. Table 4.2 shows those terms (first line), as well as the expressions (following lines) that, connected with **AND** operators, form the final search query.

Table 4.2: Research Query Building

Research Question	What methods have been proposed for the derivation of software architecture models from requirements specifications?
Methods	(process OR method OR technique OR approach OR methodology OR framework)
Derivation	(derivation OR decomposition OR “decision-making”)
Software Architecture	(“architectural architecture” OR “architectural constraint” OR “architectural pattern” OR “architectural style” OR “architectural view” OR “architectural viewpoint” OR “architectural model” OR “architecture description language” OR “architecture analysis” OR “architectural analysis” OR “architecture documentation”)
Requirements	(requirement OR “non-functional” OR “quality attribute”)

4.1.3 Defining the search strategies

Once again, our search strategy used two complementary search methods [269]: automatic and manual. The automatic search uses the search terms from the previous section to find primary studies on electronic data sources. With the manual search we followed a forward snowball approach¹ to identify additional primary studies focusing on software architecture.

4.1.4 Selecting the research sources

The search string was run in four digital libraries, without date restrictions: IEEEExplore, ACM Digital Library, Science Direct, and SpringerLink. As already described in the previous chapter, these are the libraries covering the most important forums in Computer Science that publish software architecture works (e.g., journal papers, conference proceedings and workshop papers), including the best conferences and workshop in the research area, such as WICSA, ECSA, and QoSA.

4.1.5 Selecting studies

Inclusion and exclusion criteria were defined to help selecting the relevant studies for analysis and data extraction. We included peer-reviewed papers from journals, conferences and workshops that present methods to design software architecture (I1), relevant studies cited by authors of the papers we read during the conduction process obtained by forward snowball search (I2), and relevant studies suggested by experts on the topic of

¹There are two types of snowball approaches [128]: backward (from the reference lists) and forward (finding citations to the papers found).

research (I3) as recommended in [148]. On the other hand, we excluded informal literature (slide shows, conference reviews, informal reports), secondary and tertiary studies (reviews, surveys)² and studies from conferences, workshops and journals without peer-review (E1), duplicated studies or studies with the same content (E2), studies that did not answer the research question (E3), studies not written in English (E4), studies not available for download from the source bases and whose authors did not reply to our request (E5), and studies not meeting some quality criteria (E6). In cases of studies complementing their authors' previous work, only the most cited³ ones were selected, excluding the other studies as duplicate (E2). With respect to quality criteria, further details are given in the following section.

The application of these exclusion criteria happened in four rounds. In the first round, we analyzed all the candidate studies through title and abstract reading. Next, with the remaining candidate studies, we read the studies' full text, excluding some and included others with criterion I2. In the third round, we analyzed the candidate studies included by I2 through full-text reading. Finally, in the fourth round, we analyzed the candidate studies included by criterion I3 through full-text reading.

4.1.6 Assessing the quality of the studies

We defined four quality assessment criteria (QA1–QA4) to be considered when applying the excluding criteria E6, using an approach similar to that in [6] and based on bibliometric impact information. While QA1 uses a set of general and specific criteria (Table 4.3), QA2 uses the ranking of the publications fora, QA3 uses the papers' citations and QA4 relaxes QA3. Each of these criteria is discussed next.

QA1 is calculated using the *QualityScore* given by *eq (1)*, where the General (G) and Specific (S) assessment factors are summarized in Table 4.3. The result is a numerical quantification to rank the selected studies.

$$QualityScore = \left[\frac{\sum_{G=1}^4}{4} + \left(\frac{\sum_{S=1}^4}{4} \times 3 \right) \right] \quad (4.1)$$

The quality assessment checklist, with G and S composed of four items each and each one with a maximum score of 1, shows a weighted average, where S weights 3 times more than G, as the specific contribution of a study is more important than the general factors. Papers with an overall score ≥ 3 were considered "high" quality studies; papers with a score ≥ 1.5 and < 3 were considered acceptable ("medium" quality); and papers with a score < 1.5 were considered of lower quality and therefore excluded from further analysis. It is important to clarify that we do not evaluate the quality of the paper itself with this criterion, but only its contributions' alignment with our research questions (derivation of an architectural model from requirements).

²Excluding these type of studies is common practice [148].

³The citation numbers were collected from Google Scholar.

Table 4.3: Quality Assessment Checklist

General Items (G) = 25%	Specific Items (S) = 75%
G1: Problem definition and motivation of the study	S1: Method definition
<ul style="list-style-type: none"> - Explicit Definition (1) - General Definition (0,5) - No Definition (0) 	<ul style="list-style-type: none"> - Formal definition (1) - Semi-formal definition (0,5) - Lack of formalism (0)
G2: Research methodology and organization	S2: Architecture description
<ul style="list-style-type: none"> - An empirical Methodology (1) - A generalized analysis (0,5) - Lacks any proper methods (0) 	<ul style="list-style-type: none"> - Specifies more than one architectural viewpoint (1) - Describes 1 architectural viewpoint or 1 Architectural Description Language (0,5) - Does not specify any architectural viewpoint nor Architectural Description Language (0)
G3: Contributions of the study refer to the study results	S3: Evaluation of the study
<ul style="list-style-type: none"> - Explicit w.r.t state-of-the-art (1) - Implicit w.r.t state-of-the-art (0,5) - Not determined w.r.t state-of-the-art (0) 	<ul style="list-style-type: none"> - Formalized empirical evaluation (1) - Some informal evidences are provided (0,5) - Non-justified or ad-hoc validation (0)
G4: Insights and lessons learned from study	S4: Limitations and future implications of the study
<ul style="list-style-type: none"> - Explicit and technical description (1) - General recommendations (0,5) - Not described (0) 	<ul style="list-style-type: none"> - Formalized empirical evaluations (1) - Some informal evidences are provided (0,5) - Non-justified or ad-hoc validation (0)

QA2 rates papers according to the forums where they were published. It considers “high” for papers published in forums rated A, and “medium” for papers published in forums rated B, according CORE-ERA⁴ for conferences, and SJR⁵ for journals. QA3 rates papers according to their citations, giving “high” to papers with more than five citations and “medium” to papers with one to five citations. The citation numbers are collected from Google Scholar. QA4 relaxes QA3 by considering papers published after 2010,

⁴<http://portal.core.edu.au/conf-ranks/>

⁵<https://www.scimagojr.com/journalrank.php>

which may have fewer citations for being relatively recent⁶. In this case, papers that have potentially high relevance have at least one citation, and papers that have not been cited have potentially medium relevance. To be included in our review, a paper must obtain $QA1 \geq 1.5$ and its bibliometric impact criteria QA2-QA4 must be “medium” or higher.

4.1.7 Collecting the Data

To promote the understandability of the area and facilitate the data extraction, the research question was decomposed according to the four dimensions recommended by the NIMSAD framework⁷ [132]: Context, Benefits to the User, Content, and Validation. Each dimension originated several (sub)research questions, as shown in Table 4.4.

Table 4.4: Research question decomposition

Research questions per NIMSAD dimension	Reasoning
Context	
What is the main goal of the method?	Even if the main goal of a method is not to address the derivation of architectural models from requirements specifications, we still register it if the topic is discussed.
What are the method’s application domains?	The scope of the method must be well-defined, and the application domain helps defining it.
What is the method’s starting point?	There are many types of requirements specifications and models. We wish to know what are the requirements specification and models used as input.
Benefits to the users	
What are the method’s benefits to the users?	A method improves some aspects of software architecture or facilitates some tasks performed by a software architect. We wish to collect the method’s benefits.
What are the limitations of the methods?	All methods have benefits and limitations. We wish to collect the method’s limitations as described by their authors.

Continues on next page

⁶The data ranges used in Q2-Q4 are borrowed from [39], which, although not demanding, are a more inclusive.

⁷NIMSAD (Normative Information Model-based Systems Analysis and Design) is a framework to evaluate methods.

Table 4.4 – *Continuation from previous page*

Research questions per NIMSAD dimension	Reasoning
Content	
What is the coverage of the method with respect to (1) understanding the problem (specify the architecturally significant requirements), (2) finding a solution for the problem (build the software architecture), and (3) evaluating the solution (check the alignment between the requirements and architecture)?	The problem of creating a software architecture model is not different from solving any other problem. We wish to know if the method includes activities for these three phases.
What are the architectural viewpoints proposed by the method?	The complexity of a software architecture requires several views of the solution and different perspectives for different stakeholders. We wish to know what are the viewpoints suggested.
Does the method define a language or notation to represent the produced artifacts?	Each viewpoint is described using some notation or language. We wish to know what are these notations and languages.
What is the level of automation of the supporting tools?	Automation is a way for decreasing the effort of executing the method's activities.
Validation	
How is the method evaluated?	The maturity of the method is directly related to how it was evaluated (e.g., experimentation, case study).
How is the quality of the method's output validated?	Another indicator of the maturity of a method is the evaluation applied to the produced artifacts.

4.1.8 Reviewing the protocol

Once completed, the research protocol must be reviewed by experts external to the study. Our protocol was evaluated by three reviewers. This evaluation was based on a set of

slides explaining the study, an evaluation form (with questions for each part of the protocol), and the candidate protocol document. The three reviewers inserted their feedback in the form, and we analyzed it. The evaluation form contained the questions in Table 4.5. The maximum score for each question was 5. In general, the feedback received was very good, as the quality assessment results ranged from 75% (average 4) to 100% (average 5).

The reviewing process led to the refinement of the protocol, particularly: we removed the publication date restriction to be more inclusive with the journals and conferences chosen, and added the quality score assessment to mitigate any potential subjectivity in the evaluation of the general and specific factors. Additionally to the protocol evaluation, (i) we performed a pilot to identify any possible issues in the application of the protocol, and (ii) we used Fabbri's quality checklist [87] as a supplementary quality step. This checklist has a number of questions to evaluate the search string, research protocol, initial selection of studies, final selection of studies, and data extraction. Although no problems in the protocol were found during the execution of the pilot and performing of the checklist, an insight was the usefulness of a bibliographic tool to facilitate the management of all the studies. Thus, the selected studies were cataloged into a bibliography management tool⁸, and the data extracted was kept in a spreadsheet workbook.

Table 4.5: Evaluation Score

#	Question	Average
1.	Do the research questions cover the specific work objective?	4.6
2.	Are the questions clear?	4.6
3.	Are the digital libraries representative of the area of study?	5
4.	Are you aware of any relevant conference or journal related to this area of study?	NA*
5.	Are the keywords in the query sufficient to achieve the mapping study objectives?	4.3
6.	Is the query complete (e.g., missing synonyms)?	5
7.	Are the Boolean operators adequate and are they used adequately in the search query?	5
8.	Is the data extraction procedure complete enough to achieve the mapping study objectives?	4.6
9.	Are the inclusion/exclusion criteria complete enough to achieve the mapping study objectives?	5
10.	Are the quality assessment criteria complete enough to achieve the mapping study objectives?	4
* <i>This is an open question where the reviewers were asked to offer a list of publication fora.</i>		

⁸Papers tool: <http://www.papersapp.com>

4.2 Conduction

The execution of the search string (Section 4.1.2) in the four digital libraries retrieved a total of 2575 candidate primary studies, which were collected and imported into the bibliographic tool. The following step was to select the primary studies by applying the inclusion and exclusion criteria, a step that decreased the number of papers to 78 relevant studies. Table 4.6 summarizes this process. The selected primary studies were read fully, while still applying the exclusion criteria, and the relevant data was extracted and added to a spreadsheet workbook previously structured as a form. Next, we applied the quality assessment approach⁹, reducing to 39 the total number of studies for final analysis. A synthesis of the data extracted from these 39 papers is described in the next section and the list of all these papers can be found in Table 4.7.

Table 4.6: Application of the filtering criteria

Criteria	IEEE	ACM	Science Direct	Springer	Snowball	Experts
I1	+188	+905	+203	+1259	+0	+0
I2	+0	+0	+0	+0	+14	+0
I3	+0	+0	+0	+0	+0	+6
E1	-11	-22	-47	-91	-0	-0
E2	-9	-104	-10	-145	-4	-1
E3	-149	-775	-134	-999	-3	-3
E4	-0	-0	-0	-4	-0	-0
E5	-0	-0	-1	-1	-0	-0
E6	-12	-1	-3	-6	-2	-0
Total	7	3	9	13	5	2

I1 - Included peer-reviewed papers.

I2 - Included relevant studies cited by authors.

I3 - Included relevant studies suggested by experts.

E1 - Excluded informal literature and secondary and tertiary studies.

E2 - Excluded duplicated studies or studies with the same content.

E3 - Excluded studies that did not answer the RQs.

E4 - Excluded studies that were not written in English.

E5 - Excluded studies that were not available for download.

E6 - Excluded studies according to the quality assessment.

⁹19 studies were excluded by QA1, four by QA2, and one by QA3.

Table 4.7: Selected studies for the mapping study on software architecture derivation methods.

Id	Paper
S1	Z. Durdik, Towards a process for architectural modelling in agile software development, in: Proceedings of the joint ACM SIGSOFT conference–QoSA and ACM SIGSOFT symposium–ISARCS on Quality of software architectures–QoSA and architecting critical systems–ISARCS, ACM, pp. 183–192, 2011.
S2	A. S. Nascimento, C. M. Rubira, R. Burrows, F. Castor, A model-driven infrastructure for developing product line architectures using cvl, in: IEEE VII Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS), 2013, pp. 119–128.
S3	G. Loniewski, A. Armesto, E. Insfran, An architecture-oriented model-driven requirements engineering approach, in: Model-Driven Requirements Engineering Workshop (MoDRE), 2011, IEEE, pp. 31–38.
S4	F. Montero, E. Navarro, Atrium: Software architecture driven by requirements, in: 14th IEEE International Conference on Engineering of Complex Computer Systems, 2009, IEEE, pp. 230–239.
S5	L. Dai, K. Cooper, Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Network, IEEE, 2005, pp. 178–183.
S6	P. Sochos, M. Riebisch, I. Philippow, The feature-architecture mapping (farm) method for feature-oriented development of software product lines, in: 13th Annual IEEE International Symposium and Workshop on Engineering of Computer Based Systems (ECBS2006), 2006.
S7	P. Petrov, U. Buy, R. L. Nord, The need for a multilevel context-aware software architecture analysis and design method with enterprise and system architecture concerns as first class entities, in: 9th Working IEEE/IFIP Conference on Software Architecture (WICSA), 2011, pp. 147–156.
S8	E. Woods, N. Rozanski, Using architectural perspectives, in: 5th Working IEEE/IFIP Conference on Software Architecture (WICSA 2005), 2005, pp. 25–35.
S9	C. Hofmeister, P. Kruchten, R. L. Nord, H. Obbink, A. Ran, P. America, A general model of software architecture design derived from five industrial approaches, Journal of Systems and Software 80 (2007) 106–126.
S10	P. Colombo, F. Khendek, L. Lavazza, Bridging the gap between requirements and design: An approach based on problem frames and sysml, Journal of Systems and Software 85 (2012) 717–745.
S11	J. Castro, M. Lucena, C. Silva, F. Alencar, E. Santos, J. Pimentel, Changing attitudes towards the generation of architectural models, Journal of Systems and Software 85 (2012) 463–479.
S12	J. Kim, S. Park, V. Sugumaran, Drama: A framework for domain requirements analysis and modeling architectures in software product lines, Journal of Systems and Software 81 (2008) 37–55.
S13	A. Casamayor, D. Godoy, M. Campo, Functional grouping of natural language requirements for assistance in architectural software design, Knowledge-Based Systems 30 (2012) 78–86.
S14	E. Ovaska, A. Evesti, K. Henttonen, M. Palviainen, P. Aho, Knowledge based quality-driven architecture design and evaluation, Information and Software Technology 52 (2010), 577–601.
S15	P. Sanchez, A. Moreira, L. Fuentes, J. Araujo, J. Magno, Model-driven development for early aspects, Information and Software Technology 52 (2010), 249–273.
S16	J. Perez, N. Ali, J. A. Carsi, I. Ramos, B. Alvarez, P. Sanchez, J. A. Pastor, Integrating aspects in software architectures: Prisma applied to robotic tele-operated systems, Information and Software Technology 50 (2008).

Continues on next page

Table 4.7 – Continuation from previous page

Id	Paper
S17	J. Castro, M. Kolp, J. Mylopoulos, Towards requirements-driven information systems engineering: the tropos project, <i>Information systems</i> 27 (2002) 365–389.
S18	J. Gonzalez-Huerta, E. Insfran, S. Abrahao, Defining and validating a multimodel approach for product architecture derivation and improvement, in: <i>International Conference on Model Driven Engineering Languages and Systems</i> , Springer, pp. 388–404, 2013.
S19	L. Chung, S. Supakkul, N. Subramanian, J. L. Garrido, M. Noguera, M. V. Hurtado, M. L. Rodriguez, K. Benghazi, Goal-oriented software architecting, in: <i>Relating Software Requirements and Architectures</i> , Springer, 2011, pp. 91–109.
S20	M. Lucena, J. Castro, C. Silva, F. Alencar, E. Santos, J. Pimentel, A model transformation approach to derive architectural models from goal-oriented requirements models, in: <i>On the Move to Meaningful Internet Systems: OTM 2009 Workshops</i> , Springer, pp. 370–380.
S21	S. Tang, X. Peng, Y. Yu, W. Zhao, Goal-directed modeling of self-adaptive software architecture, <i>Enterprise, Business-Process and Information Systems Modeling</i> (2009) 313–325.
S22	R. Chitchyan, M. Pinto, A. Rashid, L. Fuentes, Compass: composition-centric mapping of aspectual requirements to architecture, <i>Transactions on aspect-oriented software development IV</i> (2007) 3–53.
S23	P. Sanchez, L. Fuentes, A. Jackson, S. Clarke, Aspects at the right time, in: <i>Transactions on aspect-oriented software development IV</i> , Springer, 2007, pp. 54–113.
S24	L. Silva, T. Batista, A. Garcia, A. Medeiros, L. Minora, On the symbiosis of aspect-oriented requirements and architectural descriptions, <i>Early Aspects: Current Challenges and Future Directions</i> (2007) 75–93.
S25	R. Cordero, I. Salavert, J. Torres-Jimenez, Designing aspectual architecture views in aspect-oriented software development, <i>Computational Science and Its Applications-ICCSA 2006</i> (2006) 726–735.
S26	R. Wojcik, F. Bachmann, L. Bass, P. Clements, P. Merson, R. Nord, B. Wood, Attribute-driven design (ADD), version 2.0, Technical Report, Carnegie-Mellon University Pittsburgh PA Software Engineering Institute, 2006.
S27	P. America, E. Rommes, H. Obbink, Multi-view variation modeling for scenario analysis, in: <i>International Workshop on Software Product-Family Engineering</i> , Springer, pp. 44–65, 2003.
S28	F. Bachmann, L. Bass, G. Chastek, P. Donohoe, F. Peruzzi, The architecture based design method, Technical Report, Carnegie-Mellon University Pittsburgh PA Software Engineering Institute, 2000.
S29	E. Y. Nakagawa, M. Guessi, J. C. Maldonado, D. Feitosa, F. Oquendo, Consolidating a process for the design, representation, and evaluation of reference architectures, in: <i>IEEE/IFIP Conference on Software Architecture (WICSA)</i> , 2014, IEEE, pp. 143–152.
S30	D. Soni, R. L. Nord, C. Hofmeister, Software architecture in industrial applications, in: <i>17th International Conference on Software Engineering</i> , 1995. ICSE 1995., IEEE, pp. 196–196.
S31	P. Kruchten, <i>The rational unified process: an introduction</i> , Addison-Wesley Professional, 2004.
S32	H. Bagheri, K. Sullivan, Model-driven synthesis of formally precise, stylized software architectures, <i>Formal Aspects of Computing</i> 28 (2016) 441.
S33	A. Alebrahim, S. Fassbender, M. Filipczyk, M. Goedicke, M. Heisel, Towards systematic selection of architectural patterns with respect to quality requirements, in: <i>Proceedings of the 20th European Conference on Pattern Languages of Programs</i> , ACM, pp. 40, 2015.
S34	L. Tian, L. Zhang, B. Zhou, G. Qian, A gradually proceeded software architecture design process, in: <i>Software Process Workshop</i> , Springer, pp. 192–205, 2005.
S35	P. Grunbacher, A. Egyed, N. Medvidovic, Reconciling software requirements and architectures with intermediate models, <i>Software & Systems Modeling</i> 3 (2004) 235–253.

Continues on next page

Table 4.7 – Continuation from previous page

Id	Paper
S36	O. Raiha, H. Kundi, K. Koskimies, E. Makinen, Synthesizing architecture from requirements: A genetic approach, in: <i>Relating Software Requirements and Architectures</i> , Springer, 2011, pp. 307–331.
S37	K. Pohl, E. Sikora, The co-development of system requirements and functional architecture, in: <i>Conceptual Modelling in Information Systems Engineering</i> , Springer, 2007, pp. 229–246.
S38	R. F. Passarini, J.M. Farines, J. M. Fernandes, L. B. Becker, Cyber-physical systems design: transition from functional to architectural models, <i>Design Automation for Embedded Systems</i> 19 (2015) 345–366.
S39	A. Tang, H. Van Vliet, Software architecture design reasoning, in: <i>Software Architecture Knowledge Management</i> , Springer, 2009, pp. 155–174

4.3 Reporting: Study results

This section starts with a summary of the demographic data for the primary studies and proceeds to discussing the results according to the four NIMSAD dimensions.

4.3.1 Demographic data

Among the various types of publications, 12 are conference papers ([S1, S2, S4-S8, S18, S24, S25, S29, S30]) and 12 are journal papers ([S9-S17, S32, S35, S38]) both corresponding to a total of 61.4% of the selected primary studies. From the total number of primary studies, 7 are book chapters ([S19, S22, S23, S31, S36, S37, S39], accounting for 17.9% of the studies), 6 are workshop papers ([S3, S20, S21, S27, S33, S34], corresponding to 15.3%) and, finally, 2 are technical reports ([S26, S28], or 5.1% of the total number of selected studies).

4.3.2 Context

What is the main goal of the method? From the 39 selected studies, 35 (89.74%) address the derivation of software architecture [S1-S4, S6-S8, S10-S29, S31, S33-S39], 3 (7.7%) concern the understanding of how an architectural model is created in practice [S9, S30, S32], and one (2.56%) [S5] targets design and analysis of quality attributes [S5]. Although most of the studies discuss software architecture derivation, their focus differs significantly, as described in Table 4.8.

It is important to notice that, although there are some supporting tools for the derivation process (e.g., QuaDAI [S18] and Monarch [S32]), all the decisions reported in the selected primary studies were based on the architects’ tacit knowledge (meaning that they are currently based on the experience and intuition of the authors), and the quality of the artifacts were evaluated subjectively, when evaluated at all (this will be further discussed in in Section 4.3.5, under the question *How is the method evaluated?*).

Table 4.8: Context of the selected studies.

Selected Study	Context
S1	Relationship between agile methods and architectural modeling.
S2, S6, S12, S18, S28	Use software product line concepts to derive a software architecture.
S2, S3, S4, S11, S14, S15, S18, S38	Use model-driven techniques to derive an architectural model.
S5, S7, S14, S15, S19, S18, S26, S33, S36	Focus on the satisfaction of non-functional requirements at architectural level.
S8, S27, S31	Describe different views to represent the complex software architectures.
S10, S33	Create software architecture from a problem frame specification.
S11, S17, S19, S20, S21, S37	Derive a software architecture from organizational goals specifications.
S13	Introduce an approach for mining and grouping functionalities (architectural modules) from textual descriptions of requirements using text mining techniques
S16, S22, S23, S24, S25	Use aspect-oriented techniques to derive an architectural model.
S29	Show a process for the design, representation, and evaluation of reference architectures.
S34, S35, S39	Facilitate the software architecture design. Help architects during architectural design

What are the method's application domains? From the total number of studies, only one focuses on a specific domain (cyber-physical systems) [S38]. The remaining 38 studies (97.4%) do not discuss the methods' fitness (or unfitness) to particular application domains. Instead, the authors limit their approaches to a given paradigm or technological platform (e.g., model-driven development [S2, S3, S4, S11, S14, S15, S18, S38], aspect-oriented software development [S16, S22-S25], software product lines [S2, S6, S12, S18, S28]). Regarding examples or case studies used for evaluation, 20 studies were illustrated with case studies [S2, S4, S6, S7, S10, S11, S13-S17, S20, S22, S24, S25, S29, S32, S35, S38, S39], 10 use examples [S1, S3, S5, S8, S19, S21, S23, S33, S36, S37], 3 report experiments

for specific application domains [S12, S18, S36], and 7 were not illustrated [S9, S26-S28, S30, S31, S34].

What is the method’s starting point? The goal is to identify the type of requirements and requirements models used as a starting point. Table 4.9 summarizes the results, that we discuss next. All the 39 studies address functional requirements, as expected, and from those, 30 (76.9%) address non-functional requirements [S4, S5, S7-S12, S14-S30, S33-S37]. To aggregate these findings we use Sommerville’s [236] classification of **NFRs** into Product **NFRs** (specify the desired characteristics a product must have), Organizational **NFRs** (derived from organizational policies and procedures), and External **NFRs** (derived from factors external to the system and its development process). From those 30 studies, 28 (93.3%) consider product **NFRs** [S4, S5, S7-S12, S14-S16, S18, S19, S21-S30, S33-S37], 9 (30%) consider organizational **NFRs** [S7, S11, S12, S17, S20, S26, S28, S30, S37], and one (3.3%) considers external **NFRs** [S7]. Some of the studies consider more than one type of NFR, resulting in a sum of the percentages by type of **NFRs** greater than 100%. In particular, 7 studies (23.3%) analyze product and organizational **NFRs** [S7, S11, S12, S26, S28, S30, S37], and one study (3.3%) considers product, organizational and external **NFRs** [S7]. Figure 4.1 (left) depicts the distribution of the studies that consider **NFRs** as input, also showing the overlaps among types of **NFRs**.

Table 4.9: Requirements Types and Models Used as Input, where ✓ means requirements type/model used and ✗ means not used.

#	Functional	Product NFRs	Organizational NFRs	External NFRs	Textual	Goal-oriented	Unspecified	Problem frames	Feature model	Sequence model
S1	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗
S2	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗
S3	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗
S4	✓	✓	✗	✗	✗	✓	✗	✗	✗	✗
S5	✓	✓	✗	✗	✗	✗	✓	✗	✗	✗
S6	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗
S7	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗
S8	✓	✓	✗	✗	✗	✗	✓	✗	✗	✗
S9	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗
S10	✓	✓	✗	✗	✗	✗	✗	✓	✗	✗
S11	✓	✓	✓	✗	✗	✓	✗	✗	✗	✗
S12	✓	✓	✓	✗	✗	✓	✗	✗	✗	✗
S13	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗
S14	✓	✓	✗	✗	✓	✗	✗	✗	✗	✗
S15	✓	✓	✗	✗	✗	✗	✗	✗	✗	✓
S16	✓	✓	✗	✗	✗	✓	✗	✗	✗	✗
S17	✓	✗	✓	✗	✗	✗	✓	✗	✗	✗
S18	✓	✓	✗	✗	✗	✗	✗	✗	✓	✗
S19	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗
S20	✓	✗	✓	✗	✗	✓	✗	✗	✗	✗
S21	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗
S22	✓	✓	✗	✗	✓	✗	✗	✗	✗	✗
S23	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗
S24	✓	✓	✗	✗	✗	✓	✗	✗	✗	✗
S25	✓	✓	✗	✗	✓	✗	✗	✗	✗	✗
S26	✓	✓	✓	✗	✓	✗	✗	✗	✗	✗
S27	✓	✓	✗	✗	✓	✗	✗	✗	✗	✗
S28	✓	✓	✗	✗	✓	✗	✗	✗	✗	✗
S29	✓	✓	✗	✗	✗	✗	✓	✗	✗	✗
S30	✓	✓	✓	✗	✓	✗	✗	✗	✗	✗
S31	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗
S32	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗
S33	✓	✓	✗	✗	✗	✗	✗	✓	✗	✗
S34	✓	✓	✗	✗	✓	✗	✗	✗	✗	✗
S35	✓	✓	✗	✗	✓	✗	✗	✗	✗	✗
S36	✓	✓	✗	✗	✓	✗	✗	✗	✗	✗
S37	✓	✓	✓	✗	✓	✗	✗	✗	✗	✗
S38	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗
S39	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗

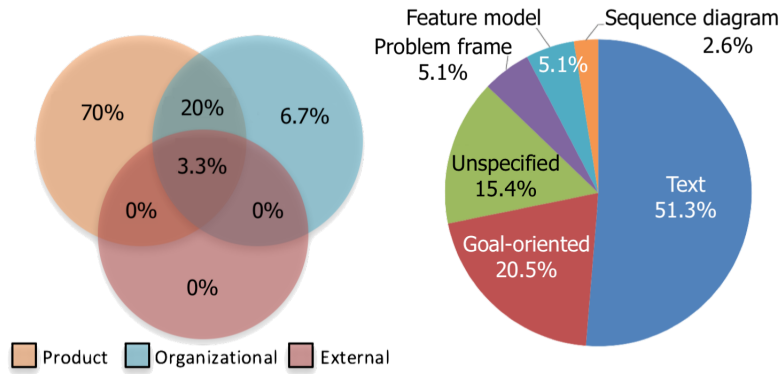


Figure 4.1: Distribution of NFRs (left) and requirements models (right).

Another relevant aspect is the requirements model used as input, summarized in Table 4.9. A total of 20 studies (51.3%) use textual requirements, 8 (20.5%) use goal-oriented models, 6 (15.4%) do not specify any type of model used, 2 use feature models, 2 use problem frames, and 1 uses a sequence diagram model.

Textual Requirements The high number of studies using textual specifications (in 20 out of 39 studies) may be because industry still uses mostly natural language to specify requirements [S22]. From those, 9 (45%) structure their requirements using use cases [S1-S3, S28, S31, S34, S36, S37, S38], 4 (20%) use intermediary models — e.g., [S14] uses Quality Attribute taxonomy, [S22] uses Requirements Description Language (RDL) [55], [S23] uses Theme/Doc [227], and [S35] uses Component-Bus-System-Property model —, and 2 (10%) create clusters to facilitate modularization and identification of architectural components [S7, S13] ([S13] uses natural language processing algorithms, and [S7] is strongly based on the software architects' experience and intuition). Finally, 5 (25%) of the studies do not give enough information about the structure of the requirements specification [S25, S26, S27, S30, S39].

Goal-oriented models From the 8 studies using goal-oriented models, 3 use generic goal-oriented models [S4, S12, S19], 3 use i-Star [S11, S17, S20], one uses KAOS [S21], and another one uses v-graph [S24].

Unspecified models The group of 6 studies ([S5, S8, S9, S16, S29, S32]) that does not specify any type of model used as input for their methods share the idea that architects should analyze and define the critical architectural requirements based on their own experience and, starting with the results of that analysis, begin the architectural modeling process.

4.3.3 Benefits to the users

What are the benefits the method brings to the users? From the 39 studies, 30 (76.9%) do not explicitly describe who the users are (target audience) and what are the benefits for them [S1, S2, S4-S7, S9, S11, S12, S14-S30, S32, S35-S37]. In such cases, we consider that the main benefit to the users is the proposal itself. The main benefits of the analyzed studies were the derivation of the software architecture from the requirements (33 out of 39) [S2-S4, S6-S9, S11, S14-S25, S27-S39], followed by understand the requirements specification (12 out of 39) [S3, S7, S9, S13, S22, S26, S28, S30, S31, S34, S35, S37], decision making support (12 out of 39) [S7, S8, S14, S17, S18, S19, S28, S33, S34, S35, S36, S39], architectural Knowledge reuse (6 out of 39) [S1, S8, S14, S18, S21, S25], improve the modularization of software architecture (2 out of 39) [S15, S16], and traceability between requirements and architecture (1 study) [S12]. The result is very interesting because, on the one hand, sharing knowledge between architects is difficult [S8], and, on the other hand, the architectural decision-making process is an important aspect during architectural design. Both have a significant effect on the structure and functionality of the system being developed, hence on the success or failure of the overall software project [204]. However, while systematic processes exist for software architecture model derivation, all architectural decisions have been exclusively based on the architects' tacit knowledge. Thus, further investigation on how to collect and later reuse the architect's tacit knowledge during the decision making process would be useful.

What are the limitations of the method? From the total number of studies, 32 (82%) did not clearly specify limitations or weaknesses [S1, S3-S7, S9-S20, S23, S25-S30, S32-S35, S37]. These are essential elements for building a sound body of knowledge in a research area. The reason for this lack of information may be due to the level of complexity involved, particularly in finding the balance between the many different forces playing a major role when creating a software architecture [88]. The limitations found are related to the immaturity of the approaches and lack of supporting tool [S21], poor requirements understanding processes [S36], undetailed decision making processes [S8, S31], inconsistency between the artifacts [S2, S22], semantic loss [S24], and lack of traceability between models [S2, S22].

4.3.4 Content

What is the coverage of the method with respect to (1) understanding the problem, (2) finding a solution for the problem, and (3) evaluating the solution? In general, a software architecture design approach is composed of an architectural analysis phase to understand the problem, an architectural synthesis phase to propose solutions to solve the problem, and an architectural evaluation phase to evaluate if the proposed solution solves the problem. Hence, we emphasize that the problem of creating a software architecture model is not different from solving any other problem, where, iteratively, we

understand the problem, find a solution, and evaluate that solution. When analyzing the coverage of the methods with respect to these three phases, only 8 studies (20.5%) tackle the three phases [S3, S8, S14, S29-S31, S37, S39]. Including overlaps, 30 (76.9%) aim at understanding the problem and finding a solution [S1-S4, S6-S8, S10-S17, S19, S22, S23, S26, S29-S39], 3 (7.6%) focus on finding a solution (without detailing how to understand the problem) and evaluating it [S9, S18, S28], and 6 (15.3%) focus on creating a solution [S5, S20, S21, S24, S25, S27]. Table 4.10 shows the coverage of each selected study. Besides the identified gaps, the software architecture methods provide only a coarse-grained description of the proposed method, hence difficult to replicate. This may be because they are currently based on experience and intuition of the authors.

Table 4.10: Coverage of the methods, where ✓ means phase covered and ✗ means phase not covered.

Study	Understanding the problem	Finding a solution for the problem	Evaluating the solution	Study	Understanding the problem	Finding a solution for the problem	Evaluating the solution
S1	✓	✓	✗	S21	✗	✓	✗
S2	✓	✓	✗	S22	✓	✓	✗
S3	✓	✓	✓	S23	✓	✓	✗
S4	✓	✓	✗	S24	✗	✓	✗
S5	✗	✓	✗	S25	✗	✓	✗
S6	✓	✓	✗	S26	✓	✓	✗
S7	✓	✓	✗	S27	✗	✓	✗
S8	✓	✓	✓	S28	✗	✓	✓
S9	✗	✓	✓	S29	✓	✓	✓
S10	✓	✓	✗	S30	✓	✓	✓
S11	✓	✓	✗	S31	✓	✓	✓
S12	✓	✓	✗	S32	✓	✓	✗
S13	✓	✓	✗	S33	✓	✓	✗
S14	✓	✓	✓	S34	✓	✓	✓
S15	✓	✓	✗	S35	✓	✓	✗
S16	✓	✓	✗	S36	✓	✓	✗
S17	✓	✓	✗	S37	✓	✓	✓
S18	✗	✓	✓	S38	✓	✓	✗
S19	✓	✓	✗	S39	✓	✓	✓
S20	✗	✓	✗				

What architectural views does the method use? Although the various studies use models to describe one or more architectural views, 19 (out of 39) studies do not explicitly name the views used [S1, S3-S6, S9-S13, S17, S20, S22, S24, S32-S34, S38, S39]. The remaining 20 studies describe a total of 28 different views: variability view [S2, S18], scenario view [S8, S28, S31], logical view [S8, S19, S28, S31, S35, S36], development view [S8, S14, S28, S31], physical view [S8, S28, S31], process view [S8, S28, S31], structural view [S14, S15, S21, S23], behavioral view [S14, S15, S21, S23], deployment view [S14, S29], internal view [S16], external view [S16], micro view [S17], macro view [S17], functional view [S18, S27], quality view [S18], transformation [S18], modular view [S25, S26, S27, S30], component-connector [S2, S25, S26], allocation view [S26], customer view [S27], application view [S27], conceptual view [S27, S29, S30], realization view [S27], crosscutting view [S29], runtime view [S29], execution view [S30], requirements [S37], and architectural [S37].

A closer analysis of these views suggests that the same view names were used for different purposes. For example, the *development view* was used by [S8] to describe the software modules and their communications and by [S14] to describe the mapping of the software to the hardware. The contrary has also been found, that is, different view names for the same purpose (e.g., [S2] describes the *component-connector view* with the same purpose of the *development view* in [S8]). This and other similar cases seem to indicate lack of a common understanding. To make the analysis possible, we chose the Kruchten’s 4+1 Views approach [161] as a baseline method to classify the various views found, “normalizing” those views that shared either the meaning or the name. Although we could have chosen any of the existing proposals, we opted for the widely known 4+1 Views approach that details five different, but complementary, views to represent a software architecture. Table 4.11 shows the mappings and a rational for each decision, and Figure 4.2 shows the result of this analysis, for a total of 80 instances of views found and mapped to Kruchten’s 4+1 views.

Table 4.11: Mapping primary study views to Kruchten 4+1 views; when authors tackle more than one view, we address each sequentially.

Study	Views in primary studies	4+1 Views	Observations
S1	<not named>	Development view	The study describes the organization of software modules using architectural styles, so the mapping to Development view.
S2	Variability view and component-connector view	Not mapped and development view	The variability view shows the common features among different software products. The 4 + 1 approach does not contain this view. The component-connector view describes the organization of software modules (thus, development view).
S3	<not named>	Process view and logical view	The study discusses information flows (hence mapped to process view) and supports functional requirements through class diagrams (hence, logical view).

Continues on next page

Table 4.11 – *Continuation from previous page*

Study	Views in primary studies	4+1 Views	Observations
S4	<not named>	Process view and development view	The study addresses information flows through sequence diagram (process view) and describes the organization of software modules (development view).
S5	<not named>	Logical view	The study describes a logical view with different abstraction levels. The first one abstraction defines the internal behavior of an object with state chart diagrams and another one supports the functional requirements through class diagrams.
S6	<not named>	Not mapped, process view and logical view	The study has a variability view (so not mapped), represents the information flows through collaboration diagrams (process view) and supports functional requirements with class diagrams (logical view).
S7	Micro and macro views	Not mapped and not mapped	The study describes a macro and a micro architecture views, showing the context of software architecture.
S8	Development view, process view, physical view, logical view, and scenario view	Development view, process view, physical view, logical view, and scenario view	The study uses the 4+1 approach.
S9	<not named>	Not mapped	The study does not describe details of the architecture that is generated by the method.
S10	<not named>	Process view and development view	This study describes two views. The first shows the information flows using activity diagrams (process view), and the second shows the organization of software modules using a component-connector diagram (development view).

Continues on next page

Table 4.11 – Continuation from previous page

Study	Views in primary studies	4+1 Views	Observations
S11	<not named>	Development view, process view, physical view, logical view, and scenario view	The study uses the 4+1 approach.
S12	<not named>	Development view	The study describes the organization of software modules through the choice of architectural styles.
S13	<not named>	Development view	The study shows the software modules extracted from textual requirements.
S14	Structural view, behavioral view, deployment view, and development view	Development view, process view, physical view, and not mapped	The study does not give details about its development view.
S15	Structural view and behavioral view	Development view and process view	The structural view describes the organization of software modules using component-connector diagram (development view) and the behavioral views shows the information flows through sequence diagram (process view).
S16	Internal view and external view	Not mapped and development view	The internal view is related to source code (not mapped) and the external view shows the organization of software modules using component-connector diagram (development view).
S17	<not named>	Development view	The study shows software modules organization using a component-connector diagram (development view).

Continues on next page

Table 4.11 – *Continuation from previous page*

Study	Views in primary studies	4+1 Views	Observations
S18	Variability view, de-functional view, quality view, and transformation view	Not mapped, development view, not mapped, and not mapped	Variability view describes the common features among different software products. Quality view represents the important quality attributes of system. The transformation view is related to model-driven techniques where a model can be transformed to other type of model. These three views do not map to any of the 4+1 views.
S19	Logical view	Development view	The view shows the software modules organization with a component-connector diagram (development view).
S20	<not named>	Development view	The study shows software modules organization with a component-connector diagram (development view).
S21	Structural view and behavioral view	Development view and process view	The structural view describes the organization of software modules using component-connector diagram (development view) and the behavioral views shows the information flows through state chart diagram (process view).
S22	<not named>	Development view	The study shows software modules organization using a component-connector diagram (development view).
S23	Structural view and behavioral view	Development view and process view	The structural view describes the organization of software modules using component-connector diagram (development view) and the behavioral views shows the information flows with sequence diagram (process view).

Continues on next page

Table 4.11 – *Continuation from previous page*

Study	Views in primary studies	4+1 Views	Observations
S24	<not named>	Development view	The study shows software modules organization using a component-connector diagram (development view).
S25	Modular view and component-connector view	Logical view and development view	The modular view supports the functional requirements through class diagram (logical view) and the component-connector view shows the organization of software modules (development view).
S26	Modular view, component-connector view, and allocation view	Process view, development view, and physical view	The modular view describes the reasoning about the system information flow (process view), component-connector view shows the organization of software modules (development view), and allocation view shows how software elements will be allocated to hardware elements (physical view).
S27	Customer view, application view, functional view, conceptual view, and realization view	Not mapped, scenario view, not mapped, development view, and physical view	The customer view is used to identify the major stakeholders who influence the system development. Functional view intends to describe the externally perceivable properties of the system. Both views are not in 4+1 approach. The application view describes the scenarios necessary to satisfy the customer's need (scenario view). The functional view shows gives an overview of all relevant features (development view). The realization view is related to the selection of hardware to satisfy the system features (physical view).

Continues on next page

Table 4.11 – *Continuation from previous page*

Study	Views in primary studies	4+1 Views	Observations
S28	Development view, process view, physical view, logical view, and scenario view	Development view, process view, physical view, logical view, and scenario view	The study uses the 4+1 approach.
S29	Crosscutting view, runtime view, deployment view, and conceptual view	Not mapped, process view, physical view, and logical view	The crosscutting view shows general information about the reference architecture, such as terms/concepts and variabilities (not mapped). The runtime view shows the dynamic behavior of systems using intern block diagrams (process view). The deployment view describes the hardware (such as, server machines, database servers, and client machines) using package diagrams (physical view). The conceptual view shows the module decomposition using block definition diagram (logical view).
S30	Conceptual view, modular view and execution view	Physical view, development view, and logical view	The conceptual view gives an overview of the system showing the hardware needed to execute the software modules (physical view). The modular view describes the domain specific modules (development view). The execution view details the modules, showing their internal behavior (logical view).
S31	Development view, process view, physical view, logical view, and scenario view	Development view, process view, physical view, logical view, and scenario view	The study uses the 4+1 approach.

Continues on next page

Table 4.11 – Continuation from previous page

Study	Views in primary studies	4+1 Views	Observations
S32	<not named>	Development view	The study describes software modules organization with component-connector diagram (development view).
S33	<not named>	Logical view	The study describes functional requirements using class diagrams (logical view).
S34	Development view, process view, logical view, and scenario view	Development view, process view, logical view, and scenario view	The study uses the 4+1 approach, but does not present the physical view .
S35	<not named>	Development view	The study describes software modules organization with component-connector diagram (development view).
S36	<not named>	Logical view	The study describes the architecture through class diagram (logical view).
S37	Requirements and architectural views	Logical and development views	The study describes an architecture through class diagram (logical view) and component-connector model (development view).
S38	<no named>	Development views	The study describes the software architecture using AADL (development view).
S39	<no named>	Development views	The study describes the architecture through a component diagram (development view).

The visual notation described in [161] was also used for this mapping where, for example, the development view is represented by components (modules and subsystems) and connectors (the communications) among components.

Figure 4.2-left shows the percentage of instances of the views used by the authors. The total number of instances, for the 28 views in the 39 studies analyzed, is 80¹⁰. Therefore,

¹⁰For example, consider two studies where one describes a development view and a physical view, and the other describes a development view; the total number of instances of views is three and the number of (types of) views is two.

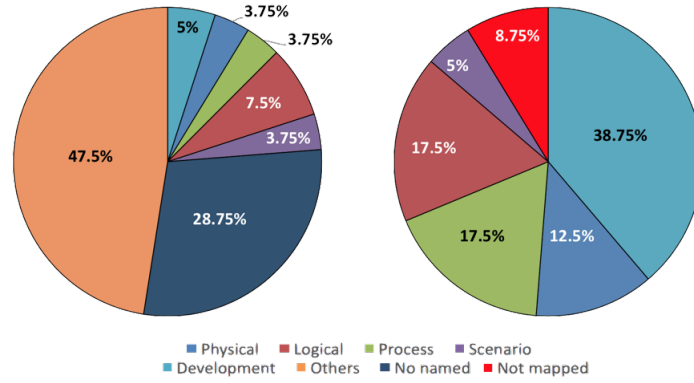


Figure 4.2: Distributions of architectural views as reported in the primary studies (left) and mapped to the 4+1 views (right).

6 out of 80 views (7.5%) are logical views, 4 are development view (5%), followed by physical, process, and scenario views with 3 views each (3.75% each). Besides, 23 views were not named (28.75%), and 38 (47.5%) were described using other names.

Figure 4.2-right shows that the percentage of studies addressing the logical and development views increased from 7.5% to 17.5% (from 6 to 14 views) and from 5% to 38.7% (from 4 to 31 views), respectively. Seven (8.75%) of the views in the primary studies were not mapped to the 4+1 views. These are, however, alternative and useful views, such as the variability view [S2, S18], the crosscutting view [S29], the customer view [S27], the quality view [S18], the macro view [S7] or internal views [S16].

Does the method define a language or notation to represent the produced artifacts?

We used Jamshidi's categories ("process algebra", "standards" and "others — non-standard"), to classify the studies with respect to the architectural description language used [130].

It is curious to note that in this analysis, the studies were grouped in three sets of the same size. Hence, one third (13 studies corresponding to 33.3%) use standard Architecture Description Languages - ADLs (e.g., UML, SysML, and AADL) [S2, S10, S14, S15, S18, S19, S29, S33-S36, S38, S39], another third does not explicitly use a standard ADL to model the architecture [S3-S5, S11, S16, S17, S20, S22-S25, S32, S37], and the third, does not indicate the architectural description language used [S1, S6-S9, S12, S13, S21, S26-S28, S30, S31] (None of the selected studies used a process algebra ADL). A possible reason for 26 studies not adopting nor defining an ADL could be because an ADL needs to capture design decisions judged fundamental to satisfy different stakeholder concerns [176] and also because it is difficult to capture all the different concerns with a unique language [169].

What is the level of automation of the supporting tools? Of the 39 selected studies, 18 (46.1%) are partially automated [S2, S4-S6, S10, S12-S16, S18, S19, S22, S32, S35, S36, S38, S39], 8 (20.5%) are not automated [S1, S20, S21, S23-S25, S33, S37], and 13 (33.3%)

do not mention supporting tools [S3, S7-S9, S11, S17, S26, S27, S28, S29, S30, S31, S34]. Given the strong dependency between the resulting architecture and the architect's tacit knowledge, it is not surprising that we could not find a complete, fully-fledged automated approach. An important fact found in studies offering some level of automation is that they all propose rules to transform a specific requirements into a specific architectural model. The generated model is then analyzed manually by the architect, and improved where necessary. This requires a considerable effort from the architect. Additionally, there is a lack of tools supporting the activities related to architectural decision making.

4.3.5 Validation

How is the method evaluated? The goal is to identify the types of empirical approaches used in the evaluation of the approaches proposed by the 39 selected studies. The result shows that 9 (23%) were illustrated with (in some cases simple) examples [S1, S3, S5, S8, S19, S21, S23, S33, S37], 3 (7.6%) were evaluated in experiments [S12, S18, S36], and that 7 (17.9%) were not evaluated [S9, S26-S28, S30, S31, S34]. Although the authors of the remaining 20 (51.2%) studies claim their approaches are evaluated with case studies [S2, S4, S6, S7, S10, S11, S13-S17, S20, S22, S24, S25, S29, S32, S35, S38, S39], these are, in fact, illustrative examples (or else the details of the case study are not given in the papers). The conclusion is that, in general, there is a lack of strong and rigorous empirical evidence. For example, no study describes what hypotheses are being evaluated (case study design), what data is collected, or how the data analysis is performed to check if the purpose of the case study has been reached.

A final fact indicates that 27 (69.2%) of the evaluation approaches were performed in academia [S1-S3, S5, S8, S10, S13-S15, S17-S25, S29, S32-S39], 8 (20.5%) have their roots in industry [S4, S6, S7, S11, S12, S16, S29, S35], and 2 ([S29, S35] have academic and industrial case studies.

How is the quality of the method's output validated? Despite recent studies showing that ATAM (Architecture Tradeoff Analysis Method) [143] is the most used and mature architecture assessment method [54], only 4 (10.2%) of the selected studies use it, or suggest its use [S8, S9, S19, S33]. In addition, [S15] uses ASAAM¹¹ (Aspectual Software Architecture Analysis Method [252]), [S34] uses SAAM, and S5 uses FDAF (Formal Design Analysis Framework) [64]. We found that 8 (20.5%) of the studies use different scenario-based approaches in an attempt to evaluate the architectural model early in the development¹² [S6, S11, S12, S14, S18, S28, S29, S37] and 21 (53.8%) of the studies do not inform about the evaluation method that can be used with their approaches [S1-S4, S7, S10, S13, S16, S17, S20-S27, S30-S32, S35, S36, S38, S39].

¹¹This is a method based on SAAM [142], a precursor of ATAM.

¹²Architecture defined only conceptually or with a high-level structure.

4.4 Overview of the results

The major findings discussed in some detail for Context (Section 4.3.2), Benefit to the users (Section 4.3.3), Content (Section 4.3.4) and Validation (Section 4.3.5) are summarized next.

4.4.1 Context

None of the approaches analyzed are concerned with supporting the architects' design decisions or increasing the quality of the artifacts. All of the approaches are methodology-specific (e.g., aspect-oriented). Around 51% of the methods use textual specifications as their input, and 65% of these structure the requirements using use cases or some other intermediate model. Finally, there is a lack of approaches to build architectural models considering external non-functional requirements.

4.4.2 Benefit to the users

Given that several studies address more than one benefit (e.g., [S3, S7, S8, S14-S19, S21, S22, S25, S26, S28, S30, S31, S33-S36]), the sum of the number of studies in the following analysis is larger than the total number of analyzed studies. Overall, 33 studies (84.6%) focus on the derivation of the software architecture from the requirements, 12 (30.7%) are concerned with understanding the requirements specification, another 12 aim at providing decision making support, 6 (15.3%) focus on the reuse of architectural knowledge, 2 (5.1%) aim at facilitating the modularization of software architecture, and one (2.7%) focus on providing traceability support between requirements and architecture. Although decision making support and architectural knowledge reuse are considered benefits of some studies, it is noticeable that all the studies are strongly based on the architect's experience. Only 18% of the studies acknowledge their limitations, which include immaturity of the approach, lack of supporting tool, poor requirements understanding process, undetailed decision-making process, inconsistency between the artifacts, semantic loss, and lack of traceability between models.

4.4.3 Content

A point worth highlighting is the apparent confusion and lack of standardized terms. We argue that the architecture of a software system is a complex task that cannot be described in one single model. Perhaps this is why ISO [127] does not commit itself to any particular views for software architecture description. This lack of standardized terms in software architecture reduces understandability and makes the communication between the involved stakeholders inefficient and error-prone [230]. Our analysis also shows that 66.6% of the studies did not use a standard ADL and that an effort is needed to develop supporting tools (1/3) of studies do not have or do not mention support tools. Standardization of terms would facilitate the adoption of ADLs. We also observed that

the methods can hardly be used by novices or less experienced software architects due to lack of detail, (the methods are strongly based on the architect's tacit knowledge and lack of systematization) and tool support, as only 46.1% is partially automated, 20.5% is not automated, and 33.3% do not offer information about tools.

4.4.4 Validation

Case studies is the most used mean for evaluation (mentioned in 51.2% of the studies analyzed). However, in most cases, these case studies are examples to illustrate the approaches. This state of practice should be improved with the adoption of stronger qualitative and quantitative evaluation methods. Regarding validation, 53.8% of the approaches do not provide an explicit way to evaluate the software architecture against the requirements specification.

4.5 Validity threats and their mitigation

Internal validity Due to the large number of definitions for the same concept, there is the risk of **not including relevant studies**. The terms used in this study were reviewed by external reviewers as recommended in [148]. We also performed a pilot study to validate the use of the terms and applied snowballing to add relevant studies cited by the authors of included approaches. We used the test-retest strategy on a random subset of selected studies to assess consistency of the selection process as recommended in [148]. Additionally, we included studies suggested by experts, as also recommended in [148]. Regarding the mapping study, we performed two evaluations: we analyzed if the results found in the pilot execution were consistent with the results of the final implementation, and we asked the three external reviewers to check, for a subset of papers, if there were interpretation problems. Subjectivity could be another threat happening during **planning and execution** due to the length of this study. We were so totally immersed in the work that objectivity could be thought as an issue. To mitigate this, we used three external reviewers to evaluate all the phases of the protocol. Additionally, we used Fabbri's best practices checklist [87] to check our work. A final threat concerns the **quality** of the selected papers for analysis and data extraction, as there is no agreement of how this task should be performed. To mitigate this risk we chose only peer-reviewed papers, used the QualityScore approach to reduce the subjectivity of the analysis, and used quality assessment criteria based on bibliometric impact information (approach widely used in systematic reviews published in the literature).

External validity We could have **missed venues** (e.g., conferences, journals) with relevant published works or have not analyzed relevant articles due to their unavailability. To avoid the first issue, we did not restrict our searchers to venues where more work related to our research was found. We searched in four major digital libraries for relevant related

work, and the external reviewers assessed whether these libraries were sufficient for our study and suggested some additional venues to be considered. These venues were used to crosscheck that they were being indexed by the digital libraries. Regarding the issue of unavailable papers, it has been mitigated by requesting the articles directly to the authors through the ResearchGate website and by email. Although the number of articles not available is small (only 2), we can only analyze the studies that we can find.

Conclusion validity The main conclusion validity threat is the data collection. Since we do not know how the digital libraries' search engines work, we run the risk of getting different results for each search (even because libraries can index new articles daily). Therefore, we ran the search string and, to eliminate the possibility of changes to the list of articles returned by the digital libraries, stored the returned studies in a bibliography management tool¹³ for later analysis and data extraction. To mitigate the issue about the data extraction, we decomposed the research question according to the four dimensions recommended by the NIMSAD framework [132] (e.g., Context, Benefits to the User, Content, and Validation) and structured a spreadsheet workbook as a form in order to receive the data necessary to answer the research question, as recommended in [148]. In this way, we know precisely what we want to extract from the articles and how to store the extracted data in an organized way.

4.6 Research roadmap

The previous discussions of the results and major findings highlight several open issues, suggesting worthwhile topics for future research. In particular:

1. **Specific domains.** The studies analyzed are typically methodology-specific (see Section 4.4.1). However, paradigms and technological platforms change over time and with technological evolution. We believe that developing approaches for specific domains could be seen as a means to collect systems' resources and capabilities to create new and more complex systems, such as systems of systems.
2. **Approaches addressing a wider range of NFRs, particularly external NFRs.** Most studies about the success factors of a software development project indicate that the key critical factors lie in the external non-functional requirements, such as business values satisfaction [34, 243, 247] (see Section 4.4.1). For example, the Standish Group CHAOS reports state that most software design defects are caused by value-oriented weaknesses [247]. A related relevant topic is to take into account the NFRs (intrinsic) conflicting nature, and study the influence these conflicts have on the architectural decisions and their consequent impact on the final architectural solution. Particularly, more research is needed to investigate the relationships between external NFRs and the software architecture.

¹³Papers tool: <http://www.papersapp.com>

3. **Understand the methods' limitations.** Only 18% of studies acknowledge their limitations (see Section 4.4.2). This may be due to immaturity of the area or due to the level of complexity involved. The fact is that researchers and practitioners should be encouraged to report the limitations and weaknesses of their approaches as these are essential to build a sound body of knowledge.
4. **Standards or common understanding.** The lack of standardized terms, or at least common agreements, reduces understandability and makes the communication between the involved stakeholders inefficient and error-prone [230] (see Section 4.4.3). Initiatives such as the Software Engineering Institute catalog of different definitions for the term "software architecture" should be more encouraged [231]. This standardization of terms is fundamental for an established body of knowledge, hence beneficial to all software architecture researchers and practitioners (particularly the less experienced ones).
5. **Make explicit tacit knowledge.** Most of the methods can hardly be used by novices or less experienced architects (see Section 4.4.3). These methods provide only a coarse-grained description of the proposed method, hence making replication impossible or at least too difficult. This may be because they are currently based on the experience and intuition of the authors. Thus, authors must share the process in which a software architecture is built, by providing the steps that must be executed, together with the guidelines and heuristics required to achieve the goal of the method.
6. **Tool support.** From the analyzed approaches, 1/3 does not offer or mention supporting tools (see Section 4.4.3). Thus, there is a need to develop tool support to facilitate the architects' activities. For example, tools able to suggest the best suited patterns and styles for a given application domain, or to handle specific NFRs to support decision-making activities and decrease the dependence on experienced architects. Such tools could explore artificial intelligence techniques, for example, to create a knowledge base and investigate recommendation algorithms based on the architects' knowledge.
7. **Evaluation methods.** A total of 51.2% of the approaches indicate being evaluated using case studies. However, these case studies are relatively simple illustrations of the authors' approaches (see Section 4.4.4). This state of practice should be improved with the adoption of stronger qualitative and quantitative methods to support the evaluation.
8. **Requirements satisfaction.** More than half of the studies (53.8%) do not provide an explicit way to validate the resulting software architecture against the requirements specification (see Section 4.4.4). The existing methods must evolve to facilitate the evaluation of architectural models in the early stages of the development life cycle,

or new ones should be created. This should contribute to identify major technical risks, allowing their mitigation at a minimal cost.

4.7 Final considerations

The goal of this chapter is to provide a comprehensive overview of the existing methods for the derivation of software architecture models from requirements specifications. To achieve this, we performed a systematic mapping study to find empirical evidence about software architecture derivation methods, and classified them with respect to their context, benefits to the user, content and validation. Two important findings of this study were, on the one hand, the confirmation of the lack of approaches for deriving architectural models from early requirements specifications aligned with the organizational business values and, on the other hand, that the process of building a software architecture is strongly based on the architects' experience and intuition, what makes this activity difficult for novices. Consequently, *there is a need for a systematic, traceable and simple to use approach for the transition between requirements aligned with the organization business values and architecture design*. These findings constitute the major goal of our [Ph.D.](#) research.

A VALUE-DRIVEN FRAMEWORK FOR SOFTWARE ARCHITECTURE

The general research question proposed for this [Ph.D.](#), “*How to derive value-centred architectural models systematically?*”, searches for a value-based methodological approach to support incremental software development that better aligns with the business values of an organization. The importance of this research question was confirmed by the secondary mapping study on software architecture derivation methods, discussed in [Chapter 4](#), which confirms the need for a systematic, traceable and simple to use approach for the transition between requirements and architecture design. The current chapter presents a value-driven software architecture framework that facilitates the derivation of software architecture models aligned with the business values of an organization, therefore contributing to address this work research question. This framework is supported by model-driven development techniques and is composed of three main methods, each one supported by a proof-of-concept tool.

5.1 Framework’s structure

According to the Cambridge dictionary, a framework is a “*supporting structure around which something can be built*”; or “*a system of rules, ideas, or beliefs that is used to plan or decide something*” [208]. Thus, by definition, a framework is a large-scale design [135] which serves as a guide, a sketch or overview of interlinked activities to facilitate an approach towards accomplishing a specific goal [63]. In the context of this work, we created a framework with a set of methods, processes, concepts mappings, and guidelines to provide the required structure to support the early stages of value-based software development, particularly business modeling and the derivation of requirements models and a reference software architecture. This framework, summarized in [Figure 5.1](#), is

composed of three core modules — Business Value Modeling, Agile Reference Architecture Modeling, and Goal-Driven SOA Architecture Modeling — supported by a set of [MDD](#) languages, transformations and tools developed using an Eclipse-based Implementation Environment. While the Business value modeling module focuses on building a stakeholder-centric business specification, the Agile Reference Architecture Modeling and the Goal-Driven [SOA](#) Architecture Modeling modules concentrate on generating a reference software architecture aligned with the business value specification.

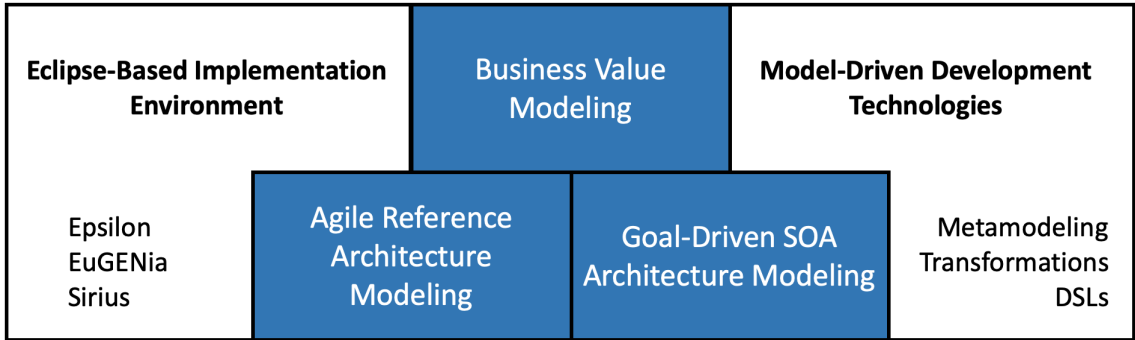


Figure 5.1: Modules, environments, and technologies of value-driven framework.

The “Business Value Modeling” module is composed of the Dynamic Value Description ([DVD](#)) method, responsible for modeling the business values of an organization, a visual technique to build value models to express the needed concepts, a modeling language implemented as using metamodeling and a [DSL](#), a step-by-step process model, a set of supporting guidelines, and conceptual mappings for a [MDD](#) derivation of a [SOA](#) capability reference architecture model. [DVD](#) uses a cognitive early requirements method aimed at analyzing and representing business values exchanged between the different parties involved in a given business activity. That is, it offers an environment wherein stakeholders can share their economic interests and views (The [DVD](#) method is further discussed in [Section 5.2](#).)

The “Agile Reference Architecture Modeling” module is composed by the method Reference Architecture Modeling for an Agile software development scenario ([RAMA](#)). [RAMA](#) is a value-centric method developed to address the architecture-agility combination. It uses the business value model created by the [DVD](#) method, and [MDD](#) and conceptual modeling techniques together with [Agile practices](#). The result is a feasible approach that combines agile development practices and software architecture design principles during the creation of the information system’s reference architecture. Due to the particular methodology followed, [RAMA](#) produces a reference architecture well-aligned with an organization business values. [RAMA](#)’s supporting modeling environment includes metamodels, semantic rules, [DSLs](#), transformations scripts, a process model and heuristics. ([RAMA](#) is discussed in more detail in [Section 5.3](#).)

The “Goal-Driven SOA Architecture Modeling” module is composed of the KAOS modeling approach for Service-oriented architecture ([KAOS4Services](#)). [KAOS4Services](#)

is a systematic approach to derive SOA applications from Goal-oriented models aligned with the business values. In other words, the DVD model is used as the start point to generate the goal-oriented model used throughout the KAOS4Services method. In contrast to existing service-oriented works that do not offer detailed and systematic methods for business analysis and services identification [20, 112, 154], KAOS4Services offers a systematic approach supported by model-driven development techniques and a set of mappings and guidelines applied to goal concepts. KAOS4Services is defined by its own modeling language, defined and implemented as a DSL, a process, a set of guidelines, and conceptual mappings to guide its use. (A detailed discussion is offered in Section 5.4).

While RAMA is suggested for Agile software development projects, KAOS4Services works better for (academic) scenarios where goal-oriented approaches are elected. Therefore, choosing between these two methods depends on the software development methodology the project under development is to follow.

The following sections describe Business Value Modeling using the DVD method, Agile Reference Architecture Modeling using the RAMA method and Goal-Driven SOA Architecture Modeling using the KAOS4Services method. The chapter ends with some final considerations.

5.2 Business value modeling

There is a widespread agreement [106, 212, 263] regarding the importance of business models for a company to express its business value, be it economic, social, environmental, technical, or other. Additionally, the success of a company's impact in the market may also depend on the alignment between its information systems and the value models expressing its economic perspective. To achieve this alignment, the value model must be used as a starting point of the software requirements specification process, guiding the software development according to the company's business economic values [36].

The Dynamic Value Description (DVD) method was developed to capture the key business values concepts through models easy to build and simple to understand by non-IT experts. This Section describes the DVD concepts and their relationships, its abstract and concrete syntax, its semantics and supporting process.

5.2.1 DVD in a nutshell

The DVD method is an early requirements specification approach whose goal is to analyze and represent the economic values exchange through a model called Dynamic Value Description (same method name). From the DVD model, both business analysts and requirements engineers specify the business values exchanges by using a cognitive-based requirements approach. The cognitive requirements approach facilitates the domain understanding because it provides an environment wherein all the stakeholders can share

their views and abstractions in a semi-structured mindmap model. In other words, the DVD model was structured to inherit the well-known characteristics from mindmap structure (e.g., simple, easy to use, useful, and accessible model [51]).

DVD is composed of six main concepts: **Actor**, **Value exchange**, who starts the value exchange, **Value port**, **Value level agreement**, and value activity. An actor is an economically independent entity. A company, business unit, role, or a customer are examples of actors. “Economically independent” refers to the ability of an actor to be profitable or to increase value for him/herself. Actors are specialized in *main actor* and *environment actor*. Each time, the business analyst focuses the analysis on the *main actor* and represents its relationship with others *environment actors*, producing an inter-organizational network. As the focus changes, the actor playing the role of “main actor” also changes. With this change in focus, new actors and *value exchanges* may appear.

A value exchange shows economic reciprocity through two *value ports*, one entry and one exit, which points to value objects (e.g., money, goods, services or information). The business analyst should define *who starts a value exchanges* and the corresponding *value activities*. While *who starts the value exchanges* aims at identifying the actor responsible for starting the value exchange, the value activities aim to specify how the value exchanges can be operationalized in a high-level of abstraction. Finally, each value exchange may have a quality *level of agreement* agreed between the involved actors. This level of agreement is a particular business aspect that must be minimally agreed among the actors in order to enable the value exchanges; it defines the business constraints based on the business strategies. For example, a company of the feeding segment provides food fresher than its competitors, as a business strategy. However, it can only guarantee “fresher food” if its suppliers deliver fresh ingredients. Therefore, the business analyst can specify a **Value Level Agreement (VLA)** defining quality constraints on the values exchanged, in this case of these ingredients. In other words, VLA is typically associated with Non-functional Requirements (**NFRs**) or Quality attributes. Non-functional Requirements (**NFRs**) refer to the operational quality of a system, as well as the constraints imposed on a solution [217]. Thus, we can define a **VLA** as a NFR at the business abstraction level.

Figure 5.2 shows a “sketch” model representing all the DVD concepts for the following example scenario: A Shopper buys low cost goods from a Store. The Shopper makes a payment and receives the good in return. In order to offer this good to the Shopper, the Store needs to acquire this same good from a specific Manufacturer. To this end, the Store places an order to receive the goods as soon as possible, aiming at refilling their stocks quickly.

The actor *Store* is the focus on the analysis. It is therefore the main actor of our DVD model — rectangle marked with “M” on the left upper corner — and *Shopper* and *Manufacturer* are the environment actors (rectangles marked with “E” on the left upper corner). *Shopper* starts a value exchange with *Store*, making a *payment* and receiving a *good* in return. The lines between actors represent the value exchanges and the black squares on the side of an actor indicate the actor that starts the relation (who starts). A

VLA associated with the value exchange between *Shopper* and *Store* is *low cost* of products. *Store* also starts a value exchange with *Manufacturer*, making an *order* aiming at receiving *goods* with fast delivery.

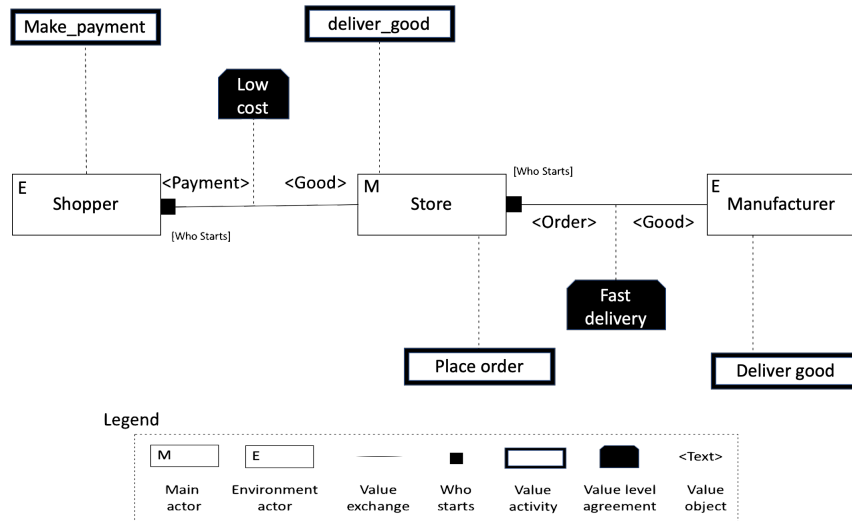


Figure 5.2: Illustration of the DVD concepts.

A DSL was defined and implemented to support the DVD concepts and relationships. Such DSL has been defined in terms of its abstract syntax (in a metamodel), a set of the rules (constraints) and its concrete syntax (visual notation). DVD is also supported by a process. All this is discussed in the following sections.

5.2.2 DVD abstract syntax and constraints

The DVD language abstract syntax is described using a metamodel. Such metamodel defines each concept with its attributes together with the relationships among the various concepts. Figure 5.3 shows this model using an ecore metamodel [47] diagram¹. Concepts are defined as metaclasses and are represented by rectangles, attributes are defined by their names and types and shown in the second compartment of the corresponding metaclass, and relationships are represented by arrows between the metaclasses. (The gray node metaclasses correspond to abstract metaclasses.)

As the DVD model is structured as a mindmap, most of the DVD concepts are specializations of the *Node* and *Relation* metaclasses, as per the mindmap metamodel defined by Siochos and Papatheodorou in [228].

In the metamodel, we created an abstract class to represent the actors (Actor class). Actors are classified (or specialized) in *main actor* (MainActor class) and *environment actor* (EnvironmentActor class). Each actor has a name (name attribute in Actor class) and a unique attribute (id attribute in Actor class). An actor is involved in at least one

¹Ecore is the core metamodel of [Eclipse Modeling Framework \(EMF\)](#) (Eclipse Modeling Framework). EMF is an Eclipse-based modeling framework and code generation facility for building tools and other applications based on a structured data model. Ecore is also its own metamodel [47], it is, it defines itself.

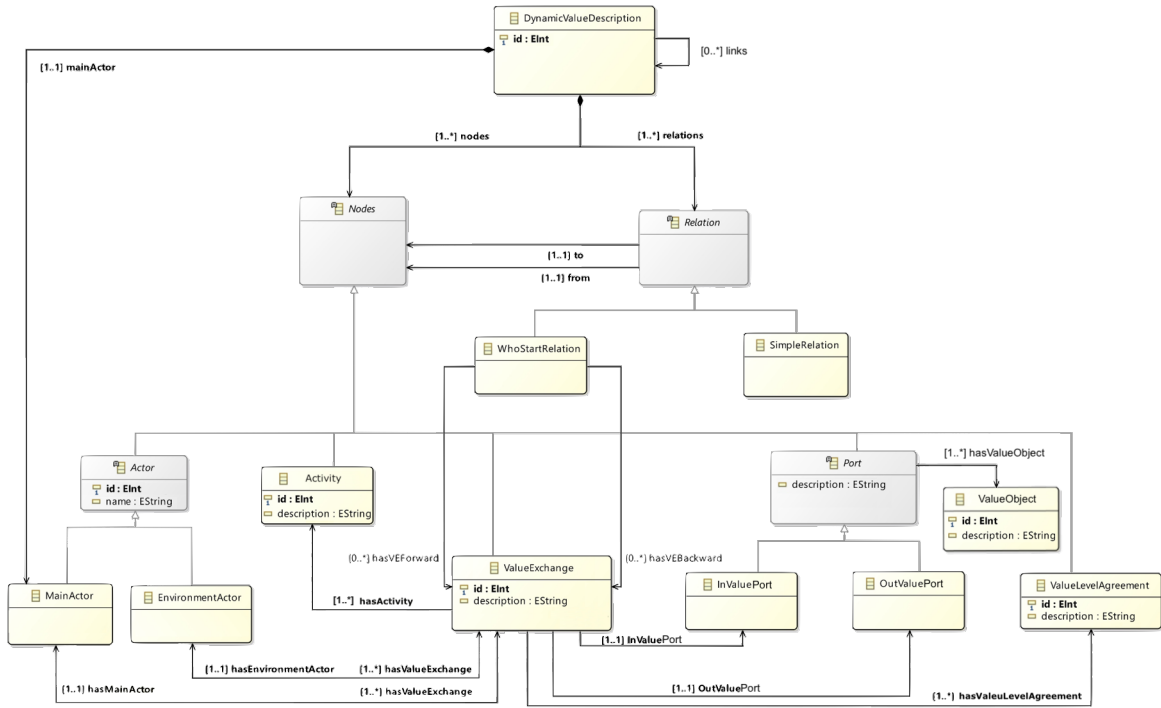


Figure 5.3: Dynamic Value Description metamodel.

value exchange (ValueExchange class) and each value exchange has at least one value level agreement (ValueLevelAgreement class) and one value activity (Activity class). Each value exchange, activity, and value level agreement has an identifier and a description. Also, the value exchange also has two value ports: the input value port (InValuePort class) and output value port (outValuePort class). Each value port has a value object (ValueObject class), a description and an identifier.

The relation presented in the metamodel are of two types: Who Starts (WhoStartsRelation class) and Simple (SimpleRelation class). Each of these has only one source node ([1..1] *from*) and a target node ([1..1] *to*). The relationship between the main actor and the environment actor is made through the WhoStartsRelation relation, having as the source node a *main actor* and as the target node a *environment actor*. In this relation, if the main actor starts the value exchange, its object is set to be *hasVEBackward* and, in contrast, if the environment actor is who starts the value exchange, then its object is placed into *hasVEForward*. Both, the *hasVEBackward* and *hasVEForward* are need to show visually who starts the value exchange. In other words, it is the visual representation of the *who starts* concept.

Each actor can be involved in more than one value exchange, but each value exchange can only belong to an environment actor. This relationship is represented by the bidirectional link [1 .. *] *hasValueExchange* and [1..1] *hasEnvironmentActor*, respectively. The same rule happens with main actor ([1 .. *] *hasValueExchange* and [1..1] *hasMainActor*). In

turn, each value exchange has an input value object, represented by the link `[1..1] InValueObject`, and an output value object, represented by the link `1..1] OutValueObject`. Each value exchange may have at least one value level agreement; this link is made through the *SimpleRelation* relation, where the source node is a *ValueExchange* and the target node is a *ValueLevelAgreement*. Also, *activities* can be defined to clarify how the value exchange can be operationalized (`[1 .. *] hasActivity`).

Certain correctness-construction rules cannot be assured by the metamodel. Those cases are defined using *Epsilon Validation Language (EVL)*² To ensure correct DVD models, a few rules we defined, particularly to guarantee:

- **Only one central node.** As the DVD is structured as a mindmap model, we need to ensure that the DVD model has only one central node. Also, the central node must be a main actor.
- **A value exchange belongs to two actors.** A value exchange is always associated with one main actor and one environment actor.
- **Economic reciprocity.** A value exchange is reciprocal by nature, that is, an actor gives something and receives in return something else. Therefore, a value exchange must have one input value port and an output value port.
- **Mandatory fields.** Actors, activities, value exchanges, ports, value objects, and value level agreements must be a unique (have an identifier) and have a name or description.

The above constraints were all implemented using EVL. As an example, Listing 5.1 shows two constraint rules performed in the context of a value exchange object. The first rule checks if a value exchange has one main actor and one environment actor associated to it. The second rule checks if a value exchange has two value ports, one input port `inValuePort` and one output port `outValuePort`.

Listing 5.1: Example of DVD semantic rule

```

1 context ValueExchange {
2
3   constraint ActorsDefinition {
4     check : self.mainActor.isDefined() and self.environmentActor.isDefined()
5     message : 'The_value_exchange_must_be_associated_to_both_'main_actor'_and_'
              'environment_actor'.'
6   }
7
8   constraint ValuePortsDefinition {
9     check : self.outValuePort.isDefined() and self.inValuePort.isDefined()

```

²EVL contributes to model validation capabilities. More specifically, EVL can be used to specify and evaluate constraints on models, metamodels and modelling technologies [74]. rules, typically associated to (context) metaclasses.


```

10     message : 'The value exchange must have been associated to one "in port" and one "out port".'
11 }
12 }

```

The remaining constraints were implemented in a similar way and are available on [DVD repository](#)³.

5.2.3 DVD concrete syntax

After the definition of the abstract syntax, with all the concepts defined in a metamodel and extra correctness rules defined in [EVL](#) for the validity rules of the concepts and relationships, we allocated one visual representation or symbol to each [DVD](#) concept and relationship. This visual representation forms DVD's concrete syntax and allows a user to use the language to create [DVD](#) models. The concrete syntax of the [DVD](#) language is shown in Figure 5.4.

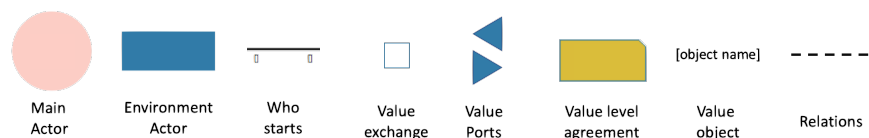


Figure 5.4: Concrete notation for the DVD language.

Taking our illustrative example from Figure 5.2 and using the [DVD](#) concrete syntax from Figure 5.4, the final [DVD](#) model would take the form of the diagram in Figure 5.5. The exception is the *value activity* concept, which we decided not to represent it in the keep the graphical representation simpler.

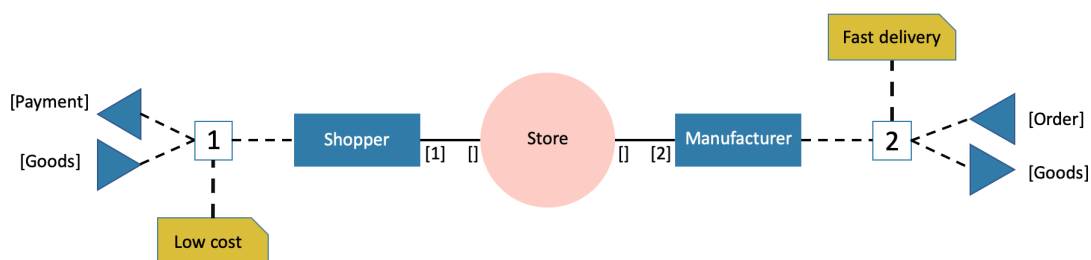


Figure 5.5: Example of a DVD model for the purchase and sale of goods by a store expressed using the DVD concrete syntax.

Note that we have changed the geometric form of the main actor to a circle aiming at passing the idea (subliminally) of a target or point of analysis. Also, we created a more expressive graphic element for the exchange of a value (a square with its identifier in the center) and to make clear the value ports used we have chosen arrows. The representation of “who starts” the value exchange was also improved in relation to the first notation

³DVD GitHub Repository: <http://bit.do/e2sYv>

where it was not possible to identify the origin of the value exchange, particularly if more than one value exchanges between two actors exists. In the current notation, we used the value exchange identifier between brackets on the line between the main actor and environment actor; the identifier must be displayed next to the originator actor. For example, the “[1]” next to Shopper indicates that this environmental actor initiates the value exchange with the identifier 1.

5.2.4 DVD process

The most notorious activity of the **DVD** method is the use of its DSL to create a correct DVD model. Such model is created following a set of steps, or activities, more rigorously described in BPMN [263], as depicted in Figure 5.6. The creation of a **DVD** model involves six activities: Identify main actor, Identify environment actors, Define value exchanges, Define who starts each value exchange, Define value level agreement, and Change the main actor. Each of these activities is explained next with excerpts of the **DVD** model example depicted in Figure 5.5.

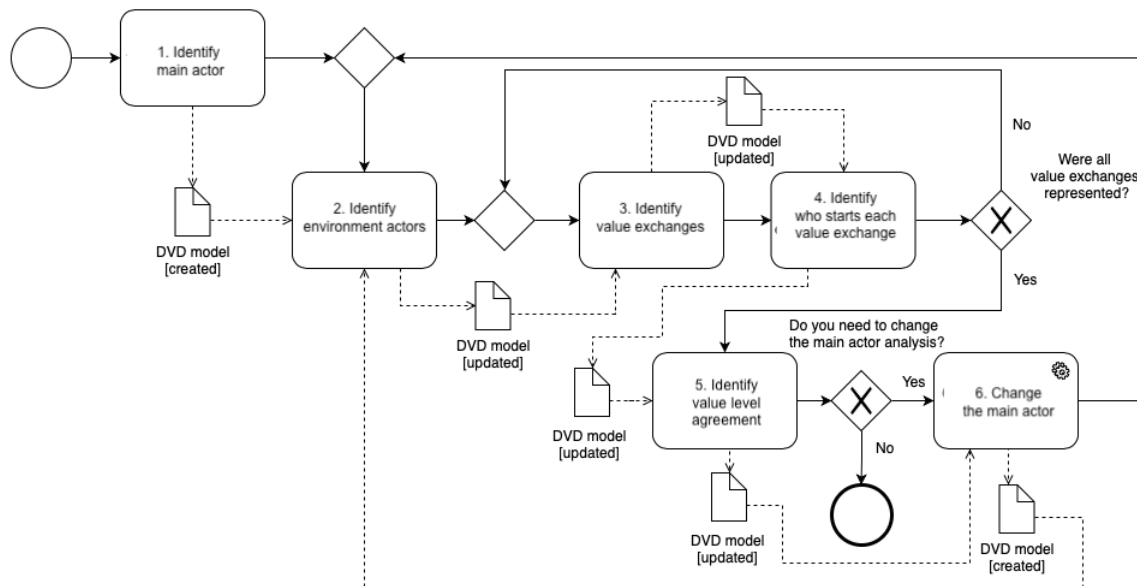


Figure 5.6: Process to create a DVD model.

1. Identify main actor. This activity starts a **DVD** model by representing the focus of the analysis. In the example, the main actor is Store, obtained by applying the following guideline.

- *Guideline 1.1:* Choose your focus by selecting the main actor for the **DVD** model; this is the actor for which you are interested in studying its value exchanges. In general, there is no need to understand the value exchanges of the entire business environment. Thus, focus only on the actor(s) (for which the information system will be developed).

2. Identify environment actors. Environment actors are those that directly interact with the main actor. The process is incremental, so we specify one environment actor at a time. The environment actors in our example are Shopper and Manufacturer and were identified using the guideline that follows.

- *Guideline 2.1:* Create environmental actors with similar level of abstraction. For example, if the identified actors are companies, then continue analyzing which other companies exchange value with this main actor. However, if the environmental actors are roles (e.g., purchasing manager), then continue analyzing the environmental actors at this level of abstraction.

3. Identify value exchanges. These add the values exchanged between the main actor and its environment actors by defining the value object related to each value port. For example, the environment actor Shopper makes a Payment to the main actor Store in exchange for a Good. In this case, Payment is represented in the output port of the value exchange (arrow heading out, as it comes from the Shopper towards the Store) and the Good in the input port (arrow heading in, as it “given” to the Shopper). In addition, business analysts can specify the activities required to operationalize the value exchanges (value Activities). These activities are not shown graphically on the [DVD](#) model. Instead, they are registered as properties of the value exchanges in the [DSL](#). The next five guidelines systematize this process.

- *Guideline 3.1:* Use products and services mentioned in the business description. All companies should offer a set of services or products to the market and/or the main actor business parties. It is through these services and products that the companies remain alive and profitable.
- *Guideline 3.2:* Use the economic reciprocity idea to find value objects. When the environment actor Shopper makes a Payment to the main actor Store, the shopper will want to receive something in return from the store (e.g., goods).
- *Guideline 3.3:* Make sure the actor is an end consumer. According to the consumer value theory [121], in general, actors who are final consumers do not aim for profit. Instead, they want to meet their needs. Although they might not aim for profit, the objects continue to have an important value for these actors.
- *Guideline 3.4:* Place the value object on the correct value port. Always consider what is received (in value port) and offered (out value port) from the point of view of the environment actor. For example, the Manufacturer receives an Order (in value port) and offers Goods in return (out value port).
- *Guideline 3.5:* Activities are needed for a business to deliver value. The operational activities of a company do not need to be specified. In general, the operational activities are automated by third parties.

4. Identify who starts each value exchange. This identifies the actor starting the value exchange. For example, the value exchange between the environment actor Shopper and main actor Store starts when the shopper makes a payment. If the shopper does not pay the store will not deliver the good and so the value exchange will not occur.

- *Guideline 4.1:* Use the value activities as the basis of an analysis to identify who starts the value exchange. Order the activities and identify who is the actor who is responsible for the first activity of value activities ordered.

5. Identify value level agreement. This defines a contract for the value exchange. So, we must understand the business constraints related to each value exchange. In the example, the requirement “low cost” of goods leads to the exchange of value between Shopper and Store. In other words, if goods were expensive, the value exchanged buyer and store would not happen.

- *Guideline 5.1:* Identify the minimum requirements needed to the value exchange to happen. Analyze why actor A makes a value exchange with actor B instead of choosing a competitor (actor C). For example, the main actor Store always places orders in environment actor Manufacturer because the Manufacturer has a fast delivery service.

6. Change the main actor. This activity allows the analyst to change the focus of his analysis to another actor. In other words, if the user selects an environment actor to become a main actor, a new DVD model is generated, and the context of that environment actor is copied to the new environment.

- *Guideline 6.1:* Set an environment actor as a main actor only when you are interested in knowing its value exchanges with other initially peer environment actors, or new environment actors down the business chain for that partner actor. In general, there is no need to understand the value changes of the entire business environment.

5.2.5 From a DVD model to a SoaML capability model

DVD is the starting point of the whole use of our framework. That is, any of the subsequent methods to generate a reference architecture, start from a DVD model. We can, however, generate a SoaML capability model using model driven techniques. In order to achieve that, we need to define a Model-to-Model transformation taking the DVD model as the source model and SoaML as the target model. This is an optional activity that serves to generate a modular description of a business in terms of its desired business outcomes.

As described in Section 3, a capability represents the ability of a business, service, or system to act and produce a result that achieves a specific goal. In this way, a capability is presented as a cohesive set of functions or resources that a the business needs to offer or to

receive. Since a capability model represents hierarchical capacities, we begin by creating an abstract capability only to be the parent of all capacities (aggregator of all capacities). Soon after, we create the second level of capabilities (children of abstract ability), where each value exchange on the DVD is transformed to a capacity in the SoaML model. The reason for this transformation is that the business needs to be able to realize the value exchanges, in order to be successful in the market. By the end, we create the third level of capabilities by transforming each activity from a value exchange into a capability in the SoaML model.

Listing 5.2 shows the implementation of the rules to transform a DVD model into a SoaML capability model using Epsilon Transformation Language (ETL). Lines 11-14 show the creation of a high-level capability aiming at being the parent of all capabilities. Next, the *value exchange* element from a DVD model is transformed into an abstract *capability* element in SoaML (Lines 18-22) and then is linked to the highest-level capability (Lines 26-29). Lines 33-37 show the DVD *activity* element being transformed to a SoaML *capability* element and Lines 42-45 show the creation of the links between the last capabilities created.

Listing 5.2: M2M Transformation: from a DVD model to a SoaML capability model

```

1 createCapability() {
2   var soamlTrans : new SoaML_Capability!SoaML_Capability;
3   var valueExchangeList = dvd!ValueExchange.allInstances();
4   var mainActor = dvd!MainActor.allInstances().first();
5   var capability;
6   var relation;
7   var abstractCapability;
8
9   //Create capability aggregator (parent)
10
11  var parentCapability : new SoaML_Capability!Capability;
12  parentCapability.name = "General_capability";
13  parentCapability.transformedFrom = "Main_Actor_ID:"+mainActor.idMainActor;
14  soamlTrans.hasCapability.add(parentCapability);
15
16  // Create abstract capabilities from dvd.valueExchanges
17
18  for(valueExchange in valueExchangeList){
19    abstractCapability : new SoaML_Capability!Capability;
20    abstractCapability.name = valueExchange.description;
21    abstractCapability.transformedFrom = "Value_Exchange_ID:"+valueExchange.
      id;
22    soamlTrans.hasCapability.add(abstractCapability);
23
24    // create a relation between capabilities
25
26    var parentRelation : new SoaML_Capability!Relation;
27    parentRelation.from = parentCapability;

```

```

28     parentRelation.to.add(abstractCapability);
29     soamlTrans.hasRelation.add(parentRelation);
30
31     // Create SoaML_Capability.Capability from dvd.activity
32
33     var activityList = valueExchange.hasActivity;
34     for(activity in activityList){
35         capability : new SoaML_Capability!Capability;
36         capability.name = activity.description;
37         capability.transformedFrom = "Activity_ID:"+activity.id;
38         soamlTrans.hasCapability.add(capability);
39
40         // create a relation between capabilities
41
42         relation : new SoaML_Capability!Relation;
43         relation.to.add(capability);
44         activity.from = abstractCapability;
45         soamlTrans.hasRelation.add(relation);
46     }
47 }
48 }

```

Figure 5.7 shows the result of applying the above transformation to the DVD model in Figure 5.5. Note that the capability names are not represented graphically in the DVD model but as attributes of the concepts. For example, the *value exchange* between *shopper* and *story* has the description “Sell products”. This description is what we use for the capability generation in the SoaML model. (The same is true for activities.)

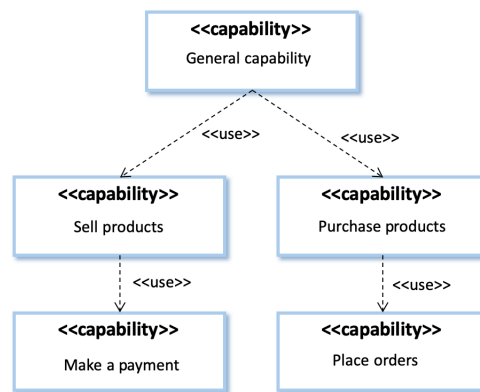


Figure 5.7: SoaML capability model generated from a DVD model.

5.2.6 About the DVD implementation

A DSL for DVD was implemented initially using the Eclipse EuGENia tool [79], allowing the creation of models syntactically validated⁴. EuGENia is a tool that automatically

⁴DVD GitHub Repository: <http://bit.do/e2sYv>

generates the background models needed to implement an Eclipse **Graphical Modeling Framework (GMF)** editor from a single annotated Ecore metamodel. However, the open-source community was no longer evolving the tool or correcting existing errors. In addition, there is little documentation to use as a knowledge base during the implementation of a DSL. For this reason, we decided to replace the EuGENia tool by the Sirius tool [80]. Sirius has been created by Obeo [187] and Thales [253] to provide a generic workbench for model-based architecture engineering that could be easily tailored to fit specific needs⁵. Basically, Sirius is supported by the same model-based technologies used by the EuGENia's tool (e.g., metamodeling, transformation between models, and model validation). The difference is that Sirius offers a lot more for its the users (e.g., documentations, forum for developers, and an active community).

5.3 Agile reference architecture modeling

Traditional software architecture processes tend to introduce excessive documentation and additional development effort of possibly unneeded features [2]. This may be why combining software architecture design and agile development was ranked second in the top “ten burning research questions” for the agile community [95]. However, combining these topics is challenging [2, 267], as proved by a recent study [267] showing that the architecture-agility combination still lacks supporting techniques. The first challenge is the apparent mismatch between the architectural design development plan and the fact that agile practitioners do not pay much attention to planning, favouring handling changes during development instead of designing early a well-structured architecture [267]. The second challenge points to valuable information being lost or misunderstood due to communication issues between business and software developers, leading to wrong or needless architectural features [267]. This section addresses the above issues by proposing **RAMA**, a value-centric development method; it starts with an overview of agile practices and follows by detailing a method for software architecture modeling for agile development.

5.3.1 An overview of the supported agile concepts

Agile development aims at reducing the effort-intensive tasks in software development, focusing on fast response to the various changes in a project [83]. The Agile Manifesto establishes values and principles to guide the agile development [93]. In recent years, researchers and practitioners have proposed several agile practices [175], which have been catalogued by the agile alliance in its “subway map to agile practice” [9], as can be seen in Figure 5.8.

Next, we describe briefly the meaning of the Agile Practices (AP#) used in **RAMA** (described in the following section) and the reason why they are used.

⁵Sirius features can be found at <https://www.eclipse.org/sirius/features.html>



Figure 5.8: Subway map to agile practice [9].

AP01 Iterations: is a timebox during which the development takes place. RAMA uses this AP to facilitate measuring the development progress and managing the scope of the changes to in the project, hence aiding implementation and reducing the associated costs.

AP02 User stories: are functional increments describing what must be developed by the team. RAMA uses user stories because they help deliver the highest value by focusing on small and immediate customer needs.

AP03 Facilitation: is any action that facilitates the development. RAMA uses this AP because the facilitator role usually focuses primarily on creating the conditions to run an effective process. Thus, a facilitator also helps to accomplish the RAMA process.

AP04 Team: is a small group of people, assigned to the same project. RAMA uses this AP because it the notion of a team entails shared accountability. In order words, the outcomes should be attributed to the entire team rather than to any individual. A small team works better than a unique person or a group of people that work individually.

AP05 Backlog: is an ordered list of items representing everything that may be needed to deliver a specific outcome. RAMA uses this AP because it explicitly shows what may need to be done.

AP06 Iterative development: is the “repetition” of the software development activities for potentially “revisiting” the same work products. RAMA uses iterative development because this AP increases the quality of the product delivered.

AP07 Incremental development: is the adding of user-visible functionality to the previous software version. RAMA uses this practice because each successive development iteration in RAMA must add user-visible value.

AP08 Ubiquitous language: is the use of the vocabulary of a given business domain, not only in discussions about the requirements for a software product, but also in discussions of design. RAMA uses this AP because the use of an “ubiquitous language” mitigates difficulties between the business experts and the development team.

AP09 Simple design: is the design that uses the practice often reduced to the acronym YAGNI (You Aren’t Gonna Need It), RAMA uses this AP because it alludes to the usual counter-arguments when a programmer tries to propose a costly design element based on its future benefits only. RAMA derives an architectural model based on the current snapshot of business values.

These are the practices that RAMA will use explicitly, as highlighted in the following sections.

5.3.2 RAMA in a nutshell

RAMA creates an information system reference architecture model aligned with the economic business values of an organization as defined in a DVD model in an intuitive, interactive, and agile (fast) manner. Starting from a DVD model, the business analyst (and/or product owner) and the development team can now initiate the identification and specification of user stories for each value exchange. To achieve this, they should define operationalization scenarios for the value exchanges. Then, the business analyst (or product owner) prioritizes the value exchanges, according to the business [Return on Investment \(RoI\)](#). Next, the business analyst and the development team, with the help of a facilitator (agile practice AP03), specify conceptual models for the user stories using mind maps (AP08). As partial conceptual models are specified separately, it is common to find the same concept in different models. The potential “concept-overlap” between different conceptual models needs to be identified. The development team identifies concept overlaps with the help of the Levenshtein distance algorithm⁶ [164]. When an overlap is found, the team decides which of the models take the responsibility for that concept. These activities are performed iteratively and incrementally (AP06 and AP07). Finally, model-driven techniques are used to generate a reference architecture with its architectural components and relationships.

⁶Levenshtein distance algorithm measures the edition distance between two words.

Next, we describe the fundamentals used to operationalize this method through a DSL (with its abstract syntax, constraints, and concrete syntax) and present its associated process.

5.3.3 RAMA abstract syntax and constraints

The RAMA metamodel is too large to be presented here as a single piece. Aiming at facilitating its comprehension, we split it into three parts. *user stories specification*, *conceptual modeling*, and *traceability support*. The *user stories specification* part specifies the concepts needed to create user stories related to the value exchanges. The *conceptual modeling* part represents all the concepts necessary to create a conceptual model structured as a mindmap. Finally, the *traceability support* part interlinks all the concepts, from the DVD model to the conceptual model as a tree model. Each of these parts are discussed next.

User stories specification. A user story is a high-level representation of a software requirement. Similarly to the DVD model, the user story model is also structured based on the specialization of the *Node* and *Relation* metaclasses, as shown in Figure 5.9. However, the user story model is not a mindmap because it has no central node. Instead, it is a hierarchical model displayed in a landscape mode. We decided to use this structure in order to reuse the BehaviorMap tool infrastructure [225] and, as a consequence, decrease the evaluation effort in the construction of the proof of concept tools.

A user story is typically described using natural language following one of several possible templates. We structure the users stories as suggested by the *Behavior-Driven Development (BDD)* practice. BDD is an agile development practice that evolved from *Test-Driven Development (TDD)*⁷ and *Acceptance-test Driven Development (ATDD)*⁸ [233]. BDD was created based on software testing practices. It uses system behaviors checking to build software with quality [183]. Behaviors are specified directly with stakeholders through BDD scenarios. BDD scenarios aim at obtaining a description of the behavior of the system, taking advantage of an active communication between the stakeholders. In other words, they are intended to describe how the system implements a feature and how it should behave given a context in the occurrence of a certain event [183].

This BDD scenarios has three main parts (or Steps): *Given* (some context) *When* (some action is carried out) *Then* (a particular set of observable consequences should happen). The full formulation of the User story can be preceded by the agent that triggers the

⁷TDD was mainly adopted in the processes of agile development, born from *EXtreme Programming (XP)* [28]. It suggests that the tests be written at the beginning of development rather than at the end as in the traditional form of software development. Thus, the programmers have control over what is working correctly in the system and consequently the quality of the code produced increases [27].

⁸ATDD is a requirements capture and verification process [156]. It was evolved from TDD, with the objective of creating tests that validate the business rules of the stakeholders. With the creation of acceptance tests, there is a validation of stakeholder needs, and development progress is made on the basis of satisfied acceptance tests.

action associated to the “when” clause. For example, “As a **Ph.D. Student**, *given* that I completed all the curricular units, *when* I ask for a certificate, *then* the academic services will print one without further requirements”.

Also, each *Step* (*Given*, *When*, and *Then*) contains a *Content*. In addition to the steps, the **BDD** scenarios have a node *As a* that associates an *agent* to the scenario. In order to relate the DVD model with the user story specification, a *value exchange* is associated with one or more user stories (*UserStory* class).

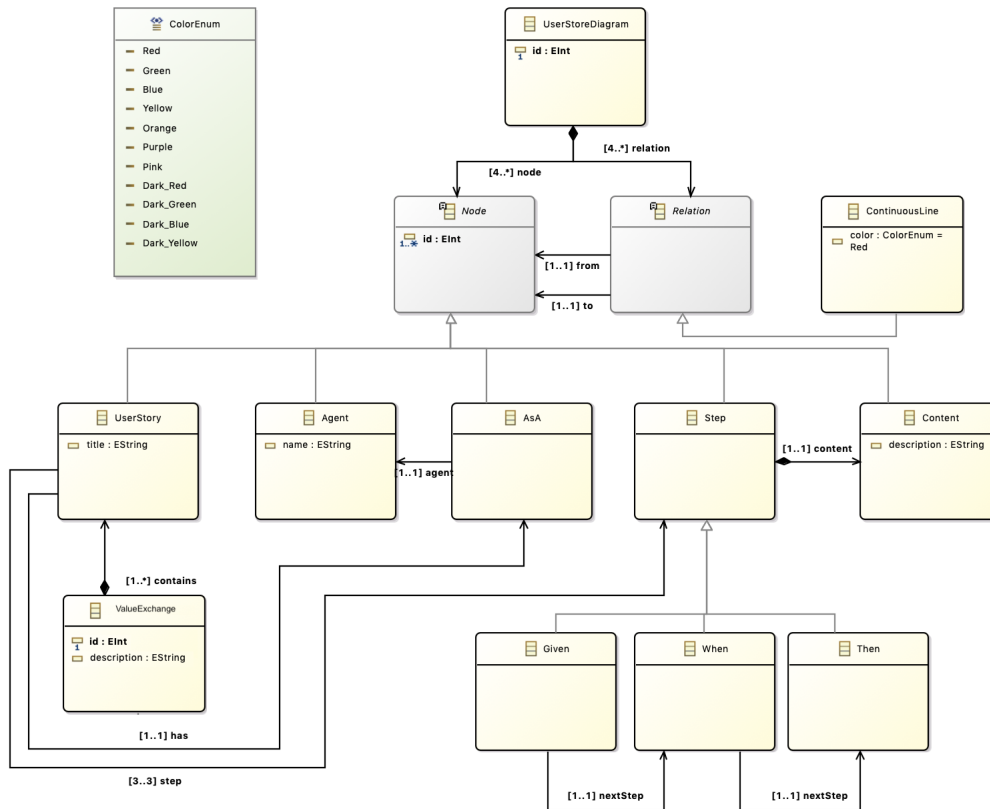


Figure 5.9: User story metamodel.

An instantiation of the user story metamodel is depicted in Figure 5.10, which shows a sketch representing the user story concepts for a *Cash withdrawal* requirement. In this illustrative example the user story is described for an agent *Bank account owner* who has a bank account in credit, and does not made withdrawals recently (*Given* step). So, *when* he attempts to withdraw an amount less than his card’s limit, *then* the withdrawal should complete without errors or warnings.

As discussed for the DVD abstract syntax, certain correctness-construction rules cannot be assured by the metamodel. Those cases are defined using **EVL** rules, typically associated to metaclasses that define the context (or scope) of the rule. To ensure correct user stories models, a few rules were defined, particularly to guarantee that:

- **All concepts must be linked.** As the user story is structured as a hierarchical model

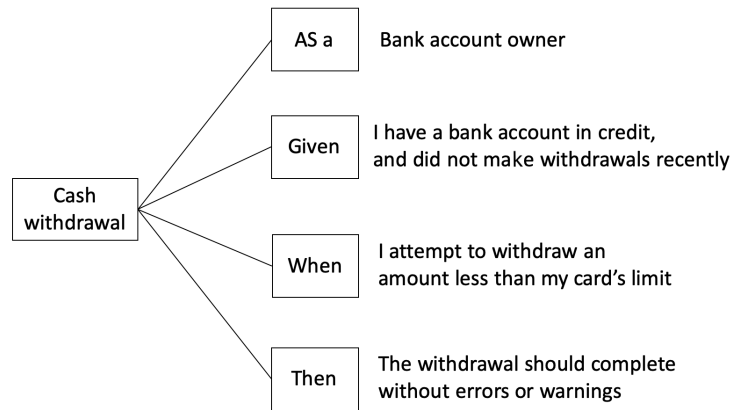


Figure 5.10: User story sketch.

in a landscape mode, we need to ensure that the concepts *Asa*, *Given*, *When*, and *Then* are always linked to the the user story node.

- **All user story must have an agent.** Any user story must specify its agent because the scenario it describes is from the point of view of that agent.
- **Steps always have a content.** All the steps of a user story must have a content. In other words, each Given, When, and Then step must have a content associated to it.
- **All attributes are mandatory.** All the attributes of all the concepts must be filled in.

Listing 5.3 shows a semantic rule where each “*given* step must have a *content*” implemented using *EVL*. All the other rules were implemented in a similar manner.

Listing 5.3: Given step must have only link to a Content

```

1 constraint GivenMustPointToContent {
2   check {
3     var n = self.nodes.select(n : Node | isGiven(n.type()) and
4       self.edges.select(c: Edge | c.source = n and
5         not (isContent(c.target.type()))).size() > 0);
6     return n.isEmpty();
7   }
8   message : 'Given_Step_must_have_a_link_to_a_Content'
9 }

```

Conceptual modeling. A Conceptual model describes “*aspects of the physical and social world around us for the purposes of understanding and communication*” [166]. A conceptual model describes the important concepts in the system domain, helping clarifying the terminology, or vocabulary, of the domain [234]. This helps the stakeholders to understand the key elements of the domain that are involved in the system being developed. We propose a conceptual model structured as a mindmap aiming at inheriting all well-known

benefits of the mindmap structure (e.g., simplicity, understandability [51]). Figure 5.11 shows the metamodel for our conceptual model. The mindmap structure is defined by the composition of *Nodes* and *Edges* corresponding to the structure of a graph. Each Node can be of type *Entity* or *Attribute*, where *Entity* is a representation of a concept from the domain and an *Attribute* is a characteristic of an *Entity*. Also, the Edges are represented as Dotted lines in our conceptual model.

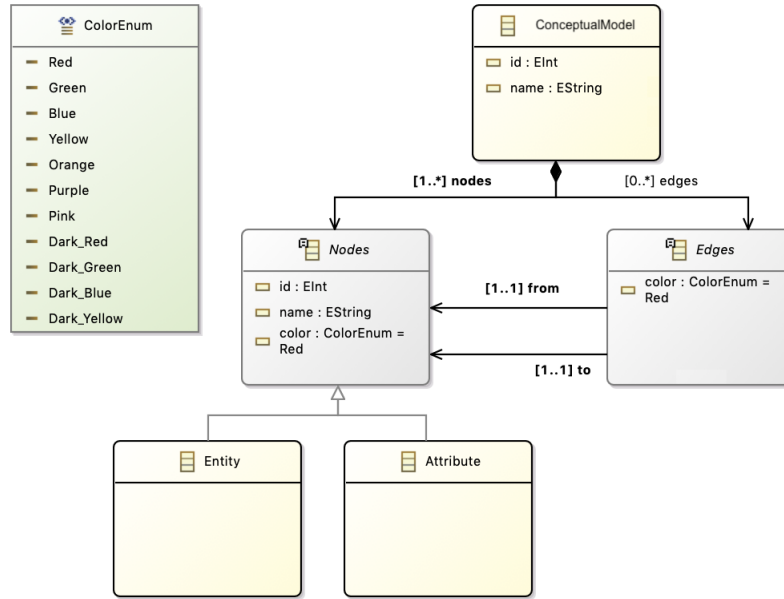


Figure 5.11: Conceptual model metamodel.

An instantiation of the conceptual metamodel is depicted in Figure 5.12, showing a sketch representing a conceptual model for a Business-to-Business e-commerce order management. In the example, a sales *Order* can be carried out by a *User* that can be an *administrator*, *operator*, or *supervisor*. This order can have different statuses (e.g., *performed*, *backorder*, and *completed* as defined in *Status*) and a *Payment* description. An order item (*OrderItem*) has an *amount*, a *price*, and a *discount* and it is related to a *Product*. A product has a *description*, a minimal *price*, and a quantity in *stock*.

To ensure that only correct conceptual models are created, similarly to the previous cases, a few rules were defined, particularly:

- **The model has only one central node.** As the conceptual model is structured as a mindmap model, we need to ensure that it has only one central node.
- **The model has a creation order.** Entities can be linked with other entities and with attributes. However, attributes cannot be linked with other attributes.
- **The entities are unique.** One entity is defined only once, i.e., if there are two entities with the same name, only one can be linked with attributes. Therefore, if more than

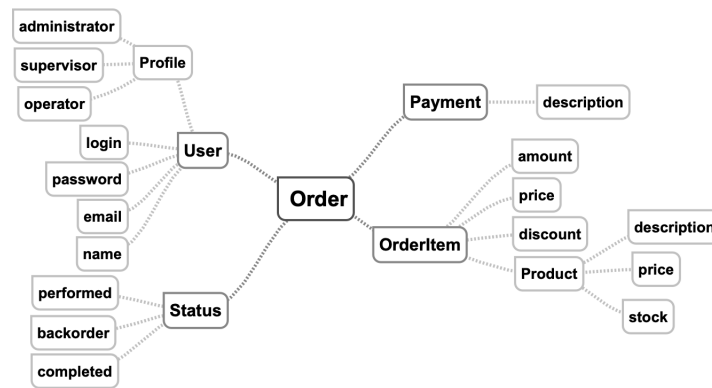


Figure 5.12: Conceptual model sketch.

one entity with the same name exist, they must be representing the same concept. So it is not good practice to scatter the (attributes) of the concept throughout different parts (entities) of the model.

- **All the attributes are mandatory.** All the attributes of all the concepts must be filled in.

Listing 5.4 shows the [EVL](#) rule to ensure that an attribute is not linked with other attributes. All other rules were implemented in a similar manner.

Listing 5.4: Attributes can only link to Entities

```

1 constraint AttributesCanOnlyPointToEntity {
2   check {
3     var n = self.nodes.select(n : Node | isAttribute(n.type()))
4     and self.edges.select(c: Edge | c.target = n
5     and (isEntity(c.source.type())
6     or isAssociativeEntity(c.source.type()))).isEmpty();
7
8     return n.isEmpty();
9   }
10  message : 'Attributes can only link to Entities'
11 }

```

Traceability support. We created a tree model to ensure traceability among the development artifacts, from the [DVD](#) value exchanges to conceptual models. Figure 5.13 shows the metamodel of this tree model, which is structured based on *Nodes* and *Edges*. This structure allows the concepts to be shown in the model in a hierarchical manner.

The model defines the various nodes are specializations of the *Node*, and a *continuous line* from the *Edge* node. A *RootNode* is the Parent node of at least one *Priority* node. *Priority* nodes have an attribute establishing their level of priority, according to the enumerated *High*, *Medium*, and *Low*, as defined in the metaclass *PriorityEnum*. A *Priority* node is parent of at least one *Value Exchange* node. A *Value Exchange* node is parent of at

least one *User Story* node. The tree model also maintains the traceability between user stories and their conceptual models ($[0..*]$ *isParent* relation from *UserStory* class to *ConceptualModel* class) and the links among conceptual models ($[0..*]$ *relatedWith* relation on the *ConceptualModel* class).

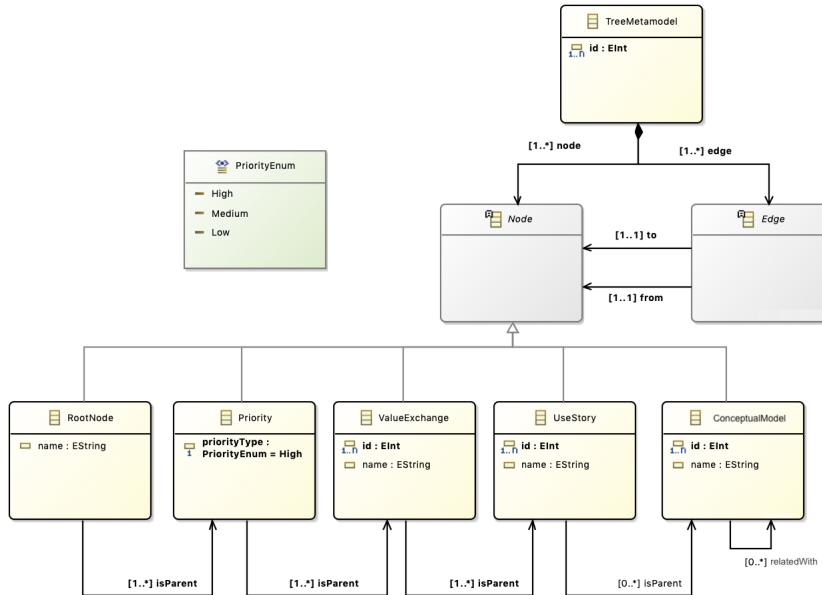


Figure 5.13: Metamodel for traceability support.

Figure 5.14 shows a sketch tree representing all the traceability supporting concepts. In this illustrative example, the top level is the root node (backlog, related to AP05), each of the three nodes in the second level represent a priority value from the priority scale used in the first-round, the third level has the value exchanges (order by the second-round priority), the fourth level has the user stories, and the last level has the conceptual models. In this way, we have a direct traceability of the value exchanges that are most important for the business and also the user stories are associated with each value exchange. Also, we are able to identify what conceptual models are associated with which user story.

To ensure a correct tree model, a few rules were defined, to guarantee the following constraints:

- **All the concepts must be linked.** As the model is structured as a tree, we must ensure that the the elements of type priority, value exchange, user story, and conceptual model are always linked among them hierarchically. In other words, there are no loose elements in the model.
- **The top of the model is always the same.** The traceability model always has a root node with three children nodes: High Priority, Medium Priority, and Low Priority.
- **The parent node is known.** The parent node of a value exchange is always a priority node, The parent node of a user story is always a value exchange node, and the

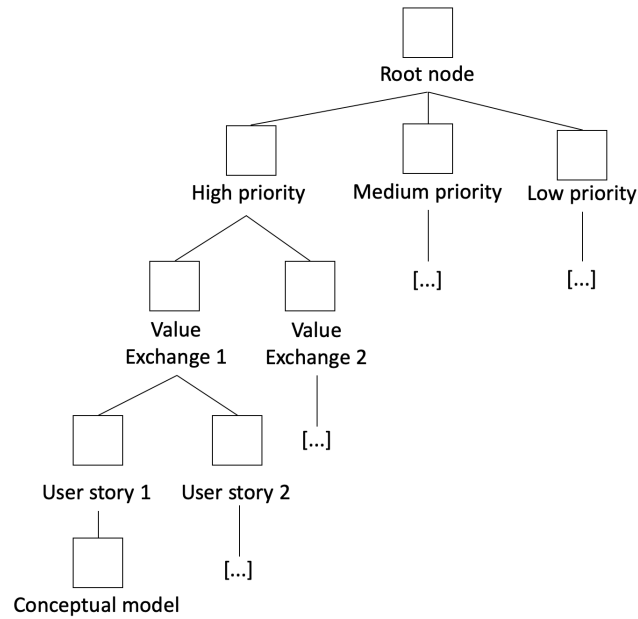


Figure 5.14: Traceability (tree) model sketch.

parent of a conceptual model node is always a user story node.

- **All attributes are mandatory.** All the attributes of all the concepts must be filled in.

Listing 5.5 shows the [EVL](#) rule to ensure that a root node has three priorities nodes.

Listing 5.5: A root node has three priorities nodes

```

1 constraint RootNodeHasThreePrioritiesNodes {
2   check {
3     var n = self.nodes.select(n : Node | isRootNode(n.type()))
4     and self.edges.select(c: Edge | c.target = n
5     and (isPriority(c.source.type())));
6
7     return n.size() == 3;
8   }
9   message : 'A Root node can only have three associated priorities nodes'
10 }

```

5.3.4 RAMA concrete syntax

User stories specification. The elements that make up the concrete syntax of a user story (inherited from the BehaviorMap tool [225]) are presented in Table 5.1.

Figure 5.15 is an instantiation of the user story model showing the Cash withdrawal example used to illustrate the concrete syntax.

Table 5.1: Concrete language for User story specification tool.

Icon	Name displayed in the editor palette	Icon	Name displayed in the editor palette
!	Given	?	When
✓	Then	📄	User Story
👤	Agent	Text (No icon)	Content

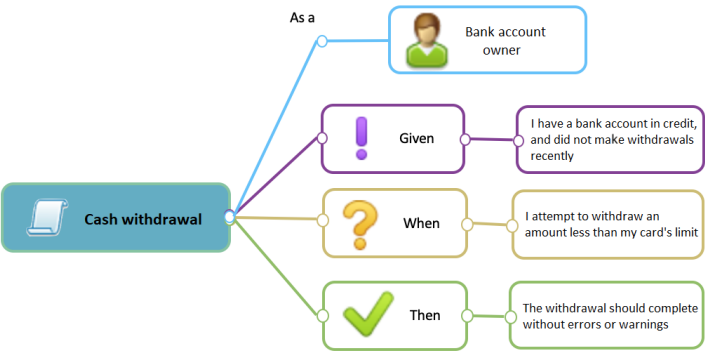


Figure 5.15: User story specification with the concrete syntax.

Traceability support. The elements that make up the concrete syntax of the traceability tree model are presented in Figure 5.16. The notation chosen for the tree elements is a circle, altering only the color according to the priority of the node [high = red; Medium = yellow; Low = blue].

Conceptual modeling. Since there are editors already available for mindmap modeling, we did not design one more language and nor did we implement one more editor. Instead, we evaluated several tools and concluded that Freemind [229], Drichards [76], DomainMap [225], and SimpleMind Lite [178] can be used for our purpose. We chose to use the DomainMap tool because it allows us to change the concrete syntax when we are creating the model.

5.3.5 The RAMA process

RAMA is more than a set of techniques with some supporting tools. The macro process in Figure 5.17 describes the whole method. It is composed of six activities: Specify user stories, Prioritize value exchanges, Specify conceptual models, Identify concepts overlaps, Decision analysis, and Create a reference architecture.

Next, each of these steps are discussed in more detail.

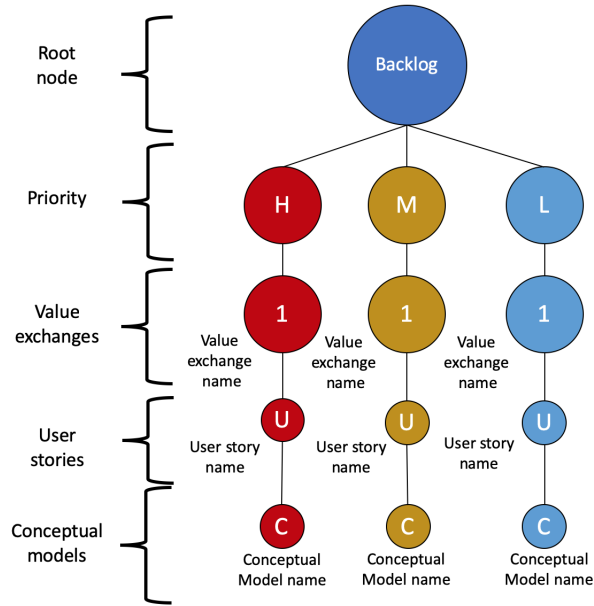


Figure 5.16: Traceability tree model concrete syntax

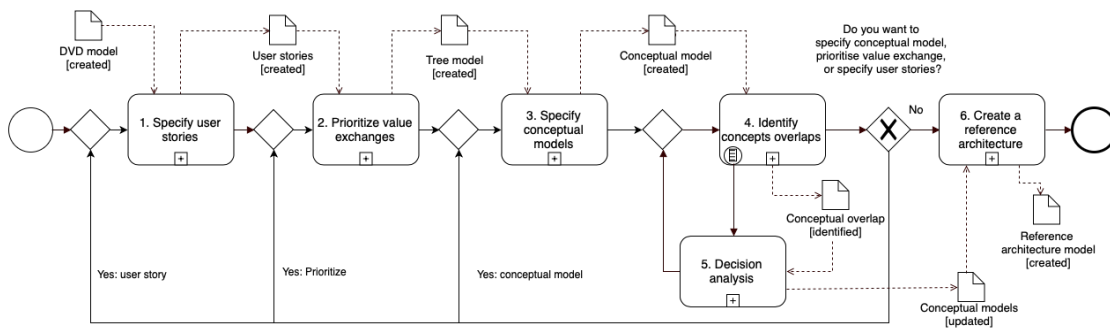


Figure 5.17: RAMA's process.

1. Specify user stories. The business person and development team define user stories (AP02) for each value exchange (or only for those with higher priority contained in an iteration). User stories describe software requirements aligned with the value exchanges which operationalize business values. We structure the users stories as suggested by [BDD](#) practice. [BDD](#) scenarios have a structure with the keywords *given*, *when* and *then* to be used in any situation as described in Section 5.3.3. That is, the keyword *given* is followed by some scenario description. The keyword *when* is followed by some action taken. And the keyword *then* is followed by some particular set of observable consequences that must be obtained after acting. To facilitate the creation of a user story, we have defined the following set of guidelines:

- *Guideline 1.1:* User stories should be defined with the help of the business analyst, to avoid the risk of writing speculative stories based on beliefs and ideas and not empirical data and evidence about the business.

- *Guideline 1.2:* Value activities can be used to facilitate the user stories specification because they describe the operationalization of the value exchange at the business level.

2. Prioritize value exchanges. Priorities are given in two rounds. In the first, the business person uses the scale high, medium, low to define the priority of each value exchange according to the [RoI](#). These priorities guide the development iterations (AP01), where the highest ranked will be implemented first. The second prioritization round, done by the development team, happens for value exchanges with the same priority (and user stories already described), to distinguish them and solving potential future conflicts. Then, it is clear which value exchanges must be handled first.

- *Guideline 2.1:* It is very important that the first prioritization is defined by the business owner or business analyst. However, the development team must also participate in the prioritization task because it improves their knowledge about the business.
- *Guideline 2.2:* The second prioritization round must be performed by the development team because they aggregate the knowledge acquired about the business with the technical expertise needed to build a solution.

3. Specify conceptual models. The development team creates conceptual models to (or part of) the value exchanges. To aid visualization and ensure traceability between value exchanges and respective conceptual models, a behavior tree view is generated from the [DVD](#) model using [M2M](#) transformations soon after the prioritization of value exchanges.

The conceptual modeling activity is collaborative, involving the business analyst, the development team, and a facilitator (who can be a member of the team) [\[260\]](#).

- *Guideline 3.1:* Armed with the value exchange specification (composed of a set of user stories), the facilitator helps the business analyst and the development team build the conceptual models, and uses a conceptual model structured as a mindmap to answer questions like: (i) What is the central concept of the problem domain? (ii) What are the sub-concepts directly related to the central concept and that are relevant to the system? (iii) What data must be managed and stored? These help eliciting relevant responses from the business analyst to build the conceptual model [\[260\]](#), which is used to aid communication. The mindmap helps to diminish the semantic gap between the business analyst and the development team [\[12, 214\]](#).

4. Identify concepts overlaps. The development team must search for similar concepts among conceptual models, identifying possible overlaps. The searches are done basically through a search of the existing text in each node of the conceptual models. To make

this comparison, we use the well-known Levenshtein algorithm [164], shown in Listing 5.6. The Levenshtein distance is a string metric for measuring the difference between two sequences, in other words, it returns the number representing the distance between two words. When the words are considered similar, it is presented a semi-automatic step to ask the user if the words are similar, or not. This semi-automatic step is discussed in the next activity.

Listing 5.6: Levenshtein Distance method

```

1 public static int levenshteinDistance(CharSequence str1, CharSequence str2) {
2     int[][] distance = new int[str1.length() + 1][str2.length() + 1];
3     for (int i = 0; i <= str1.length(); i++)
4         distance[i][0] = i;
5     for (int j = 1; j <= str2.length(); j++)
6         distance[0][j] = j;
7     for (int i = 1; i <= str1.length(); i++)
8         for (int j = 1; j <= str2.length(); j++)
9             distance[i][j] = minimum(
10                 distance[i - 1][j] + 1,
11                 distance[i][j - 1] + 1,
12                 distance[i - 1][j - 1]
13                 + ((str1.charAt(i - 1) == str2.charAt(
14                     j - 1)) ? 0 : 1));
15     return distance[str1.length()][str2.length()];
16 }
```

5. Decision analysis. The development team must decide where the overlapped concepts belong, following the principles of domain-driven design [86], where the domain is modularized with a concise set of concepts guiding the software structure. A software component encapsulates a cohesive set of system functionalities. Those functionalities handle a set of entities. Those entities are represented as concepts in a conceptual model. Then when defining the conceptual model boundaries we also define the software component boundaries, leading each conceptual model to “map” to a software component in our reference architecture. So, the development team must decide if the overlapped concepts belong exclusively to component A (matching to conceptual model 1, for example) or component B (matching to conceptual model 2) or neither (a new component C).

6. Create a reference architecture. After defining the scope of each component, it is generated a reference architecture model using models transformation. In other words, the development team uses a transformation script to generate the reference architecture model. The script transforms each conceptual model in a software architecture component and creates the associations among the software architecture components. Listing 5.7 shows how the software components are created. The script reads all conceptual model entities from the tree model (line 6) and transforms each of them in a software architecture component (lines 9-13).

Listing 5.7: Rule 1: Create software components

```

1 pre{
2     var architecturalModel = new sa.SoftwareArchitectureDiagram
3 }
4
5 SoftwareArchitectureDiagram.Component::createComponents()
6 {
7     self.MindMapModel->forEach( tree:tree.MindMapModel)
8     {
9         var conceptualModel:tree.MindMapModel = new tree.MindMapModel
10        var newComponent:sa.Component = new sa.Component()
11        newComponent.name = conceptualModel.name
12        newComponent.transformedFrom = conceptualModel.id
13        architecturalModel.add(newComponent)
14    }
15 }

```

After transforming all the conceptual model entities into software architecture components, the algorithm creates the associations among these components. Listing 5.8 shows that the algorithm reads all the conceptual model entities from the tree model (line 3) and, for each conceptual model entity, it checks if there is some association (line 5). If an association exists, the algorithm reads all the existing associations (line 9), creating a software architecture association element between two software architecture components (lines 11-14).

Listing 5.8: Rule 2: Create associations

```

1 SoftwareArchitectureDiagram.Association::createAssociations()
2 {
3     self.MindMapModel->forEach( aux:tree.MindMapModel)
4     {
5         if (aux.relatedWith.size > 0)
6         {
7             var relationList <MindMapModel> = new List<MindMapModel>
8             relationList = aux.relatedWith
9             for (int i= 0; i< relationList.size; i++)
10            {
11                var newAssociation:sa.Association = new sa.Association()
12                newAssociation.from = aux
13                newAssociation.to = relationList.get(i)
14                architecturalModel.add(newAssociation)
15            }
16        }
17    }
18 }

```

The relations between components are detected through the conceptual models overlaps: if the conceptual model *A* uses a concept from conceptual model *B*, then they are related.

5.3.6 About the RAMA implementation

A **RAMA** proof of concept was implemented⁹ using the Eclipse EuGENia tool [79]. We used the first **DSL** created to the **DVD** method and added the RAMA concepts (shown in the metamodel of Figure 5.9) in the **DVD** metamodel in order to generate the **GMF** infrastructure as an editor (this editor is generated automatically by the EuGENia tool). As the user story management language uses the same structure used by the BehaviorMap tool, we have been able to reuse many of the existing semantic rules implemented in **EVL** and published in [225]. Regarding the traceability support, a **DSL** was also implemented using the same **MDD** technology. The purpose of this implementation was to make sure that we were able to generate a tree model by relating all the necessary data (value exchanges, user stories, and conceptual model). Despite the positive result, the implementation of our proof of concept **DSL** is not enough to ensure that the data is always updated. To do this, an envisioned end-user support tool for the **RAMA** method requires constant data management, using design patterns, such as the observer¹⁰ [98] to keep the data always up-to-date. About the conceptual modeling tool, we used the DomainMap [225] in our proof of concept evaluation. The DomainMap was also created using **MDD** technology in the context of a Master degree research [225], and allows us to change the visual notation of the conceptual model being created. To facilitate the conceptual overlap identification, we used the well-known Levenshtein distance algorithm [164], which measures the edition distance between two words, calculating how many operations it needs to transform a word source in another word target. Levenshtein distance algorithm was enough to the proof of concept evaluation, but we believe that future research must be performed in natural processing languages algorithms to implement an end-user support tool for **RAMA** (e.g., identification of synonymous of words).

5.4 Goal-driven SOA architecture modeling

Among the existing requirements specification techniques, goal-oriented modeling is an approach used to discover requirements based on the stakeholders needs or objectives for the system. Thus, a goal-oriented approach identifies the stakeholders' needs and decomposes them down to system's requirements. The main focus of such an approach is to analyze the "whys" (or "goals") of the software requirements and associated qualities. Because of this, goal-oriented approaches have been used aiming at improving the alignment between businesses (analyzing the stakeholders' needs of a business) and their services-oriented software [7, 20, 113, 240].

There is a wide gap between value models and goal-oriented models, because: (1) requirements engineers find it difficult to extract knowledge from value models to design

⁹RAMA GitHub repository: <http://bit.do/e2tHw>.

¹⁰The Observer pattern defines a relationship among objects so that when one object changes state, the others are notified and updated automatically [98].

information systems [238]; and (2) business specialists do not usually understand commonly used requirements techniques to express system behavior [51]. Therefore, existing service-oriented approaches do not offer systematic methods for service identification from goal models, and the principles and guidelines proposed are very difficult to follow in practice [20]. This section starts by introducing the basics of goal-oriented approaches, then presents our KAOS4Services that is a systematic approach to generate goal-oriented models from value models and to derive services from goal-models expressed using a goal-oriented model.

5.4.1 Goal-oriented approaches

Goal-oriented requirements engineering uses goals for eliciting, elaborating, structuring, specifying, analyzing, negotiating, documenting, and modifying requirements [257]. A goal-oriented model uses *goal* as the concept to provide the rationale (i.e., the *why*) for the envisioned system [257]. Several goal-oriented approaches exist, each one focusing on different activities of the early stages of information system development, offering a variety of procedures for reasoning about goals (e.g., KAOS [65], Enterprise Knowledge Development (EKD) [167], Business Motivation Model (BMM) [251], i*/Tropos [82], Goal Structuring Notation (GSN) [146], NFRs framework [57], Goal-Based Requirements Analysis Method (GBRAM) [17], Techne [41], and Goal Requirements Language (GRL) [14]). We analyzed these nine goal-oriented modeling languages to identify and align the business value concepts from DVD model with the goal-oriented concepts. Table 5.2 summarizes the existing concepts of the nine goal-oriented models analyzed.

Table 5.2: Goal-oriented concepts

#	Concept	KAOS	EKD	BMM	iStar/Tropos	GSN	NFR	GBRAM	Techne	GRL
1	Goal	X	X	X	X	X		X	X	X
2	Soft Goal	X			X		X		X	X
3	Vision			X						
4	Mission			X						
5	Objectives			X						
6	Strategies			X		X				
7	Tactics			X						
8	Operation	X	X				X	X		
9	Task				X				X	X
10	Agent	X			X			X		X
11	Actor				X					X
12	Role				X					X
13	Position				X					X
14	Organization		X	X						
15	Context					X				
16	Domain property	X								
17	Issues		X	X						
18	Obstacles	X						X		
19	Requirements	X						X		
20	Events	X								
21	Resource				X					X
22	Expectation	X								
22	Claim						X			
23	Argument		X							

We created a set of six heuristics to map the concepts between the DVD concepts and

goal-oriented concepts. Table 5.3 describes these heuristics.

Table 5.3: Heuristics (concepts map)

#	DVD concept (source)	Goal Oriented Concept (target)	Description
H1	Value exchange	Goal	A goal is an objective the system under consideration should achieve [257]. Regarding a value exchange, it is an objective the business under consideration should achieve, most likely with the use of a system.
H2	Value Level Agreement (VLA)	Softgoal	Softgoal is a goal for which there are no clear-cut criteria for whether the condition is achieved [213]. VLA refers to the minimal business rule agreed among actors with no clear-cut criteria to achieve it. Both concepts are related to the specification of quality attributes and constraints.
H3	Actors	Agent/Actor	Agents are stakeholders who interact with the system. They are responsible for achieving requirements and expectations [213]. They are sub-units of a complex social actor, each of which is an actor in a more specialized sense [134]. There are software agents and environment agents. Thus, as the DVD main actor is directly related to the information system under development, it is mapped into a software agent, and the environment actor is mapped into an environment agent.
H4	Value object	Expectation	An expectation is a type of goal to be achieved by an environment agent [213]. Therefore, when transferring a resource, the expectation is what the actor must provide to receive something in return, and the value object is what the actor provides or receives.
H5	Value object	Requirement	A requirement is a type of goal to be achieved by a software agent [213]. Therefore, when transferring a resource, the requirement is what the actor must provide to receive something in return, and the value object is what the actor provides or receives.
H6	Value object	Resource	A resource is what the actor desires to acquire [134], and the value object is what the actor provides or receives.

From this analysis resulted that **KAOS** was the approach that had a greater number of concepts related to the **DVD** model. This, together with the fact that **KAOS** has been one of the most cited goal-oriented approaches in the literature [259] and that it builds a model of the whole system, not just of part of it [213], were the reasons for choosing it for our work.

In **KAOS**, a *goal* can be defined as an intention statement about some system whose satisfaction requires the cooperation of some *agents*. Agents are either human beings (an environment agent) or automated components (a software agent) responsible for

achieving *requirements* and *expectations* [213]. A *requirement* is a low-level type of goal to be achieved by a *software agent* and *expectation* is another kind of goal to be achieved by an *environment agent*. Besides these two types of goals, requirements and expectations, KAOS also offers *behavioral goals* and *softgoal*. A behavioral goal is a high-level type of goal that needs to be decomposed until requirements and expectations are reached. A softgoal is a goal with no clear-cut criteria to be satisfied. Examples are quality attributes, such as usability. Also, an *operation* describes behaviors that agents must have to fulfill their requirements according to a functional *scenario* [213]. Thus, abstract high-level goals are iteratively decomposed into subgoals which are operationalized to operations, representing concrete functional requirements [181]. Figure 5.18 shows a KAOS model structure example.

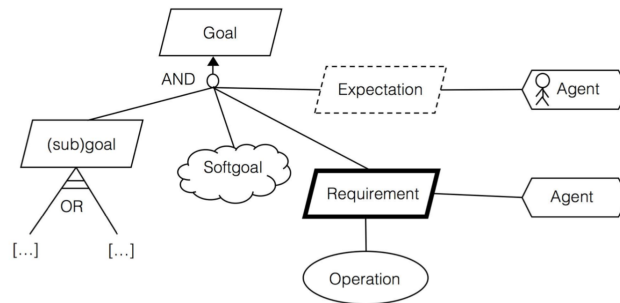


Figure 5.18: KAOS structure example from [237].

A high-level goal (root of the tree) is decomposed into (sub)goals, softgoals, requirements, and expectations using logical operators (e.g., AND and OR). And, recursively, a (sub)goal can be decomposed into (sub)goals, softgoals, requirements, and expectations using logical operators. A softgoal can be decomposed into (sub)softgoals and operations. A requirement can be decomposed into (sub)requirements and operations. Finally, an expectation can be decomposed into (sub)expectations and operations.

5.4.2 KAOS4Services in a nutshell

KAOS4Services has the purpose of removing the existing gap between value models and goal-oriented models as well as guiding the derivation of a Service-Oriented Architecture from a goal model. A SOA (Service-Oriented Architecture) development style introduces new concerns in an organization (e.g., architectural roles and development tasks), making it difficult to apply directly traditional software development approaches [20, 157]. Thus, deploying SOA in an organization requires developing an approach for services development aligned with the business concerns [20, 262]. In many SOA contexts (such as cloud computing), the business objectives are described using goal models, and these are used to define the required service compositions [11]. However, the goal-oriented requirements is not enough because they do not explicitly consider the business values. That is, a goal specification may not be aligned with the business values. Besides lack of

consensus on how service development life cycle should be conducted [112, 154], existing service-oriented works do not offer detailed and systematic methods for business analysis and services identification. Instead, they propose principles or guidelines that are very difficult to follow in practice due to lack of a systematic process [20]. This requires skilled experts to identify services and their characteristics.

This Section describes the KAOS4Services method, a systematic approach to model SOA applications using business values and goal-models. KAOS4Services generates a KAOS model from the DVD model using MDD techniques; this KAOS model is then decompose until operations are reached. With such a KAOS model, architects can generate BPMN models to better understand the business. Also, KAOS4Services offers a set of heuristics and model-driven techniques to identify services from the KAOS specification. The method classifies services into *candidate services* and *implemented services*. A candidate service is an envisioned service living in the design space and might be selected to be implemented [84]. An implemented service lives in the implementation space and is ready to be executed. KAOS4Service offers a DSL, defined in terms of its abstract syntax (in a metamodel), a set of the rules (constraints) and its concrete syntax (visual notation), and is supported by a process. The following sections discuss the various elements of the KAOS4Services method.

5.4.3 KAOS4Services abstract syntax, constraints, and concrete syntax

As happened for the previous methods (DVD and RAMA), the KAOS4Services concepts and their relationships are defined in the Ecore metamodel of Figure 5.19, forming the method's abstract syntax.

The main concepts are specializations of the (*Node*) metaclass, and the *Refinement* relationship is a specialization of the *Relation* metaclass, connecting two or more elements through the the logical operators AND and OR (defined by the attribute *logical* (set by default to AND) and which type is defined by *LogicalOperator*).

The relationship between Goals (*Goal* metaclass) are represented using the relation $[0..1] \text{ hasGoal}$ and $[0..1] \text{ hasRefinement}$. A *Requirement* can be decomposed in a *SoftGoal*, and in this case, a *SoftGoal* can only belong to a *requirement*. This relation is represented between the metaclasses *SoftGoal* and *Requirement* by the bidirectional relationship $[0..*] \text{ requireSG}$ and $[0..1] \text{ toRequirement}$, respectively.

Each *Agent* relates to *Operation* and can relate to *Requirements* (if it is a *SoftwareAgent*) or *Expectations* (if it is an *EnvironmentAgent*). A software agent may have to meet several requirements, and each requirement only belongs to a software agent, this relationship is represented by the bidirectional relation $[0..*] \text{ RResponsability}$ and $[1..1] \text{ SoftwareAgent}$, respectively. In turn, an environment agent may have to reach several expectations, and each expectation only belongs to an environment agent, this relationship is represented by the bidirectional relation $[0..*] \text{ EResponsability}$ e $[1..1] \text{ EnvironmentAgent}$, respectively. Any of the agents may have to perform several operations, but each operation only belongs

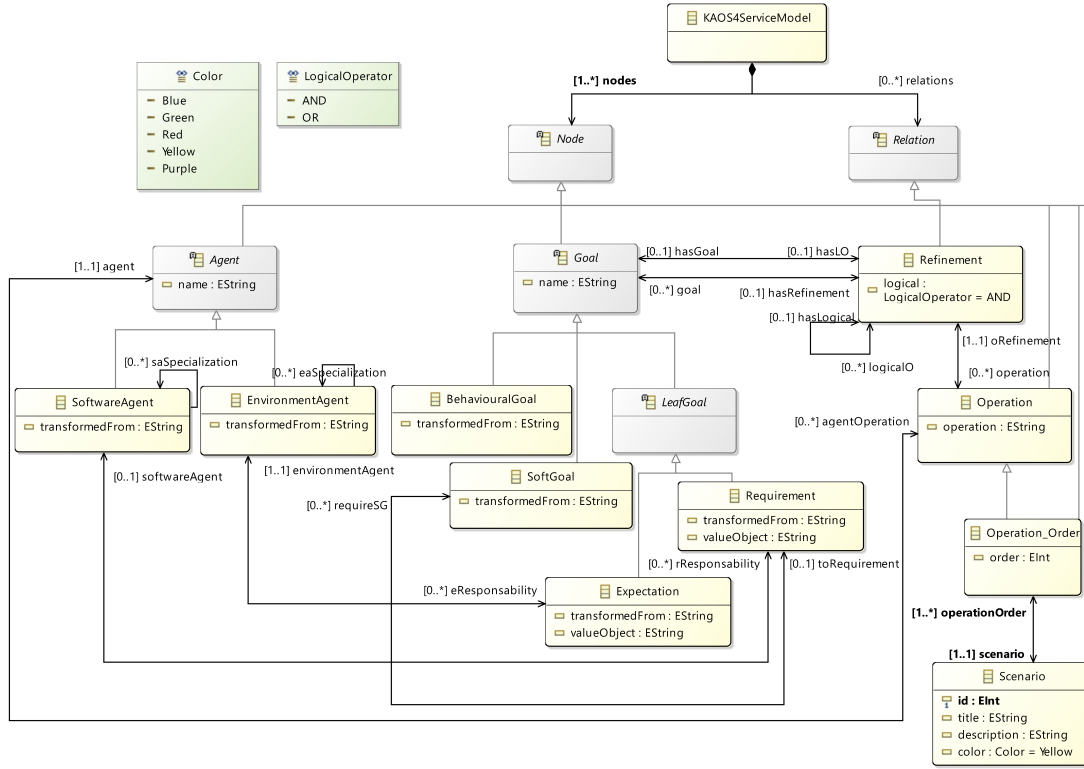


Figure 5.19: KAOS4Services metamodel.

to an agent, this relation is represented by the bidirectional relation $[0..*]$ *Operation* e $[1..1]$ *Agent*, respectively. Each agent may also specialize in other agents of the same type, i.e. one software agent may specialize in other software agents, and one environment agent may specialize in other environment agents. These relations are represented by the relation $[0..*]$ *Specialization*.

A set of operations can be related by logical operators (through the element *Refinement*), these relations are represented by the relation $[0..*]$ *ORefinement*, and by the relation $[0..1]$ *hasLogical*, which allows you to connect a *refinement* to another *refinement*. Each operation has an associated order (*Operation_Order* metaclass), and each order belongs to a single scenario, each scenario may contain a set of operation orders. This relationship is presented through the bidirectional relation $[1..1]$ *scenario* and $[1..*]$ *operationOrder*, respectively.

In order to guarantee rules of good construction of the KAOS model, in addition to the rules that are guaranteed from the metamodel, the additional semantics was defined:

- **No loose nodes in the model.** Goals, requirements, expectations, softgoals, operations, and agents must be associated with a *Refinement*. Also, a refinement links two nodes.

Unlike for DVD and RAMA, the semantic rules for the KAOS4Services method were

implemented (for each concept individually) using [Acceleo Query Language \(AQL\)](#), instead of [EVL](#). Figure 5.20 shows an [AQL](#) rule to ensure that a Goal needs to be linked to a Refinement. Similar rule was implemented for each concept of type Node (e.g. Goal). The reason for this change was the need we felt to study and analyze Sirius environment [80]. Thus, the [KAOS4Services DSL](#) was implemented in Sirius tool [80] instead of [EuGenia](#) [79] and Sirius does not support [EVL](#) (more details in Section 5.4.5).

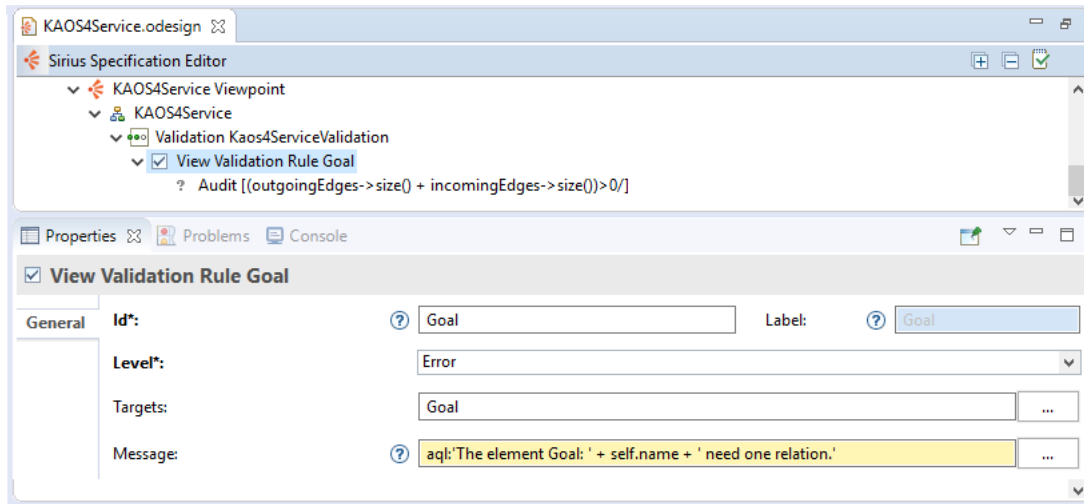


Figure 5.20: A Goal needs a relation.

The user interacts with the language through a visual notation, that is, the language concrete syntax. KAOS4Services uses a concrete syntax similar to KAOS.

5.4.4 KAOS4Services process

The [KAOS4Services](#) method uses a set of guidelines and model-driven techniques to identify services from values. This is guided by the process in Figure 5.21. This process is composed of six activities; it is iterative and incremental, but for understandability purposes its activities are displayed in cascade. Each of the activities is described next.

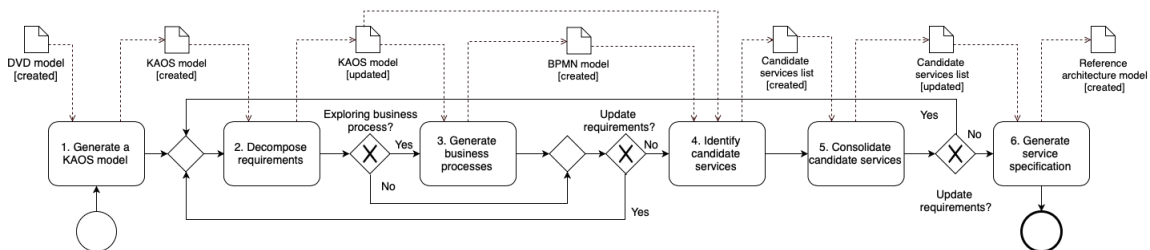


Figure 5.21: KAOS4Services process.

1. Generate a KAOS model. This first activity aims at creating a [KAOS](#) model from a [DVD](#) model. This is achieved via a M2M transformation. Listing 5.9 shows this transformation algorithm. First, the algorithm creates an abstract goal to be the parent of

goals (lines 3-10) and transforms the DVD value exchange into KAOS a behavioural goal (lines 16-19). Note that line 19 describes a relationship between the origin and the target elements, ensuring traceability between both model elements. Next, lines 22-27, create AND refinements, from the abstract goal to the behavioural goals.

Listing 5.9: Transformation from DVD to KAOS

```

1 pre {
2   "Running_ETL".println();
3   var Kaos4ServiceTrans : new KAOS4Service!KAOS4ServiceModel;
4   var initialValueExchange : new KAOS4Service!BehaviouralGoal;
5   initialValueExchange.name = "Initial";
6   var initialRefinement : new KAOS4Service!Refinement;
7   initialRefinement.logical = KAOS4Service!LogicalOperator#AND;
8   initialRefinement.hasGoal = initialValueExchange;
9   Kaos4ServiceTrans.relations.add(initialRefinement);
10  Kaos4ServiceTrans.nodes.add(initialValueExchange);
11 }
12
13 transformDVD2KAOS() {
14
15   \\ create KAOS behavioural goal
16   var valueExchange : dvd!ValueExchange;
17   var behaviouralGoal : KAOS4Service!BehaviouralGoal
18   behaviouralGoal.name = valueExchange.description;
19   behaviouralGoal.transformedFrom = "ValueExchangeID:" + valueExchange.id;
20
21   \\ create KAOS refinement AND
22   behaviouralGoal.hasRefinement = initialRefinement;
23   var refinement : KAOS4Service!Refinement = new KAOS4Service!Refinement;
24   refinement.logical = KAOS4Service!LogicalOperator#AND;
25   refinement.hasGoal = behaviouralGoal;
26   Kaos4ServiceTrans.nodes.add(behaviouralGoal);
27   Kaos4ServiceTrans.relations.add(refinement);
28
29   \\ create KAOS expectation
30   var expectation : KAOS4Service!Expectation = new KAOS4Service!Expectation;
31   expectation.transformedFrom = "OutValueObjectID:" + valueExchange.
      outValueObject.idObject;
32   expectation.valueObject = valueExchange.outValueObject.object;
33   expectation.name = valueExchange.outValueObject.description;
34   expectation.hasRefinement = refinement;
35
36   \\ create KAOS environment agent
37   var environmentAgent : KAOS4Service!EnvironmentAgent = new KAOS4Service!
      EnvironmentAgent;
38   environmentAgent.transformedFrom = "EnvironmentActor:" + valueExchange.
      hasEnvironmentActor.name + "_ID:" + valueExchange.hasEnvironmentActor.
      idEnvironmentActor;
39

```

```

40  \\ associate the KAOS environment Agent with the KAOS expectation
41  environmentAgent.name = valueExchange.hasEnvironmentActor.name;
42  expectation.environmentAgent = environmentAgent;
43  environmentAgent.eResponsability.add(expectation);
44  Kaos4ServiceTrans.nodes.add(expectation);
45  Kaos4ServiceTrans.nodes.add(environmentAgent);
46
47  \\ create KAOS requirements
48  var requirement : KAOS4Service!Requirement = new KAOS4Service!Requirement;
49  requirement.transformedFrom = "InValueObjectID:" + valueExchange.
    inValueObject.idObject;
50  requirement.valueObject = valueExchange.inValueObject.object;
51  requirement.name = valueExchange.inValueObject.description;
52  requirement.hasRefinement = refinement;
53
54  \\ create KAOS software agent
55  var softwareAgent : new KAOS4Service!SoftwareAgent;
56  var mainActor = dvd!MainActor.allInstances().first;
57  softwareAgent.transformedFrom = "EnvironmentActor:" + mainActor.name + "_ID:"
    + mainActor.idMainActor;
58  softwareAgent.name = mainActor.name;
59
60  \\ Associate the KAOS software agent with the requirements
61  requirement.softwareAgent = softwareAgent;
62  softwareAgent.rResponsability.add(requirement);
63  Kaos4ServiceTrans.nodes.add(requirement);
64  Kaos4ServiceTrans.nodes.add(softwareAgent);
65
66  \\ Create KAOS softgoal
67  var sRelation = dvd!SimpleRelation.allInstances().select(rel | rel.'to'.
    instanceOf(dvd!ValueLevelAgreement));
68  for (rel in sRelation) {
69    var verel = rel.from;
70    var vla = rel.'to';
71    if ( (verel.instanceOf(dvd!ValueExchange)) and (verel.id.equals(ve.id)) )
    {
72      var softgoal : KAOS4Service!SoftGoal = new KAOS4Service!SoftGoal;
73      softgoal.transformedFrom = "ValueLevelAgreementID:" + vla.idVLA;
74      softgoal.name = vla.description;
75      softgoal.hasRefinement = refinement;
76      Kaos4ServiceTrans.nodes.add(softgoal);
77    }
78  }
79 }

```

Next, the algorithm creates **KAOS** expectations from **DVD** value objects (lines 30-34), environment agent from environment actor (lines 37-38), and associates environment agents to **KAOS** expectations (lines 41-45). And similarly for requirements from value objects (lines 48-52), software agents from main actor (lines 55-58), and associates **KAOS**

software agents to [KAOS](#) requirements (lines 61-64). Finally, softgoals are created from value level agreements, associating these new elements with the [KAOS](#) goal generated from the [DVD](#) value exchanges (lines 67-76).

2. Decompose requirements. System requirements are decomposed into *human-intensive operations, executed by people*, and *system-intensive operations*, requiring a number of computational transactions with minimal or no human intervention [213]. Candidate services are identified from system-intensive operations, and are ordered according to the order of the operations given by the designer [109].

- *Guideline 2.1:* To facilitate the requirements decomposition into operations, identify the actions that each agent has to perform to operationalize a requirement. Each (or a set of) identified action can be mapped to one (or set of) operation.

3. Generate business processes. Although not mandatory, this activity facilitates business understanding and the services identification task, because it offers a clear step-by-step set of operations to help the architect. Model-driven techniques are used to generate [BPMN](#) business processes¹¹. The transformation is achieved by applying the following five guidelines which can be visualized in Figure 5.22. (The thick arrows indicate the guideline used to make the transformation.)

- *Guideline 3.1:* [KAOS](#) agents are transformed into [BPMN](#) pools.
- *Guideline 3.2:* [KAOS](#) operations are transformed into [BPMN](#) activities.
- *Guideline 3.3:* [BPMN](#) activities follow the order of the [KAOS](#) operations. For example, in Figure 5.22, the *Operation 2* is ordered as the first, the *Operation 3* as the second, and the *Operation 1* as the third. As a consequence, the [BPMN](#) model is created following the order: *Operation 2*, *Operation 3*, and *Operation 1*.
- *Guideline 3.4:* [KAOS](#) operations with same ordering are transformed into a [BPMN](#) parallel gateway. For example, in Figure 5.22 there are two operations with the same order (*Operation 1* and *Operation 3* have order 2)). Thus, these [KAOS](#) operations are transformed to [BPMN](#) activities between the parallel gateway (diamond with a cross (plus sign) in the center).
- *Guideline 3.5:* [KAOS](#) optional operations are transformed into a [BPMN](#) exclusive gateway. In the example illustrated in Figure 5.22, the optional operations are with their order in purple color (*Operation 2* and *Operation 3*). These optional [KAOS](#) operations are transformed into the exclusive gateway (diamond with a “X” in the center). This means that we can use either one or the other activity.

¹¹We chose [BPMN](#) because it is a standard language [263].

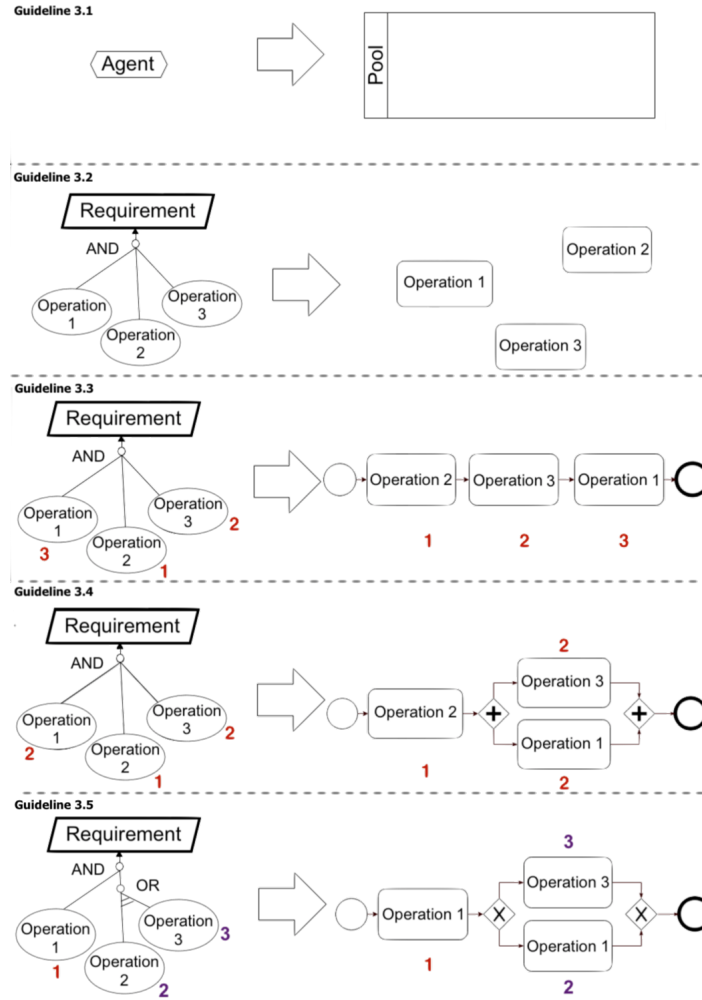


Figure 5.22: Applying the guidelines 3.1–3.5

4. Identify candidate services: The service designer selects the system-intensive operations and identifies candidate services using guidelines 4.1–4.5 which are based on workflow patterns [68, 264] and a simplified version of Nagel’s technique *et. al* [181] to order KAOS goals and operations. Each workflow pattern describes a common business behavior that can be represented through processes. Nagel’s technique uses an extended goal notation for the explicit specification of goal dependencies in a KAOS model. In other words, Nagel uses an additional textual language to complement the KAOS specification aiming at ordering the operations. Those guidelines, visually represented in Figure 5.23, are:

- *Guideline 4.1:* The *Sequence* pattern is an ordered series of activities, with one activity starting after the previous activity is completed [264]. This behavior can be defined as a “Sequential Routing” [264]. A series of sequential operations originates a candidate service.
- *Guideline 4.2:* The *Parallel Split* pattern (or *AND workflow* pattern) is a mechanism

that allows activities to be performed concurrently, rather than sequentially [264]. Operations with the same order result in an AND workflow pattern, consequently originating a candidate service.

- *Guideline 4.3:* The *Exclusive Choice* pattern (or *OR workflow* pattern) is a location in a process where the flow is split into two or more exclusive alternative paths [264]. The pattern is exclusive in that only one of the alternative paths may be chosen for the Process to continue. One OR logical operator results in an OR workflow pattern, originating a candidate service.
- *Guideline 4.4:* Conditional operations also represent an OR workflow pattern, also originating a candidate service.
- *Guideline 4.5:* Orphan operations originate a candidate service (Orphan operations are not sequential operations nor are they part of an AND pattern or an OR pattern. In most cases, they are aggregated into another candidate service.)

5. Consolidate candidate services. Service consolidation aims to help the service designers decide whether or not to implement a given candidate service. Services not selected to be implemented are not considered relevant to the system at this time and are removed from the list of candidate services. The end result is that the service designers get a list of refined candidate services. The following three service consolidation guidelines help service designers deciding about the implementation.

- *Guideline 5.1:* The sum of the number of goals the candidate service needs to achieve indicates its likelihood of being reused. We cannot define an exact goal number to affirm that from this number we can consider a candidate service because this number depends on the complexity of the system and application domain, for example. Instead, we can affirm that the greater is the number of goals, the higher will be the importance of implementing the service.
- *Guideline 5.2:* The higher the number of dependencies of a candidate service, the higher the probability of that service to be implemented.
- *Guideline 5.3:* Candidate services with a single operation (orphan operation) can be aggregated into another candidate service.

6. Generate service specification. After consolidation, a [SoaML](#) specification with all the selected candidate services is generated using MDD techniques and the following two guidelines (see Figure 5.24):

- *Guideline 6.1:* A candidate service is mapped into a [SoaML](#) service contract.
- *Guideline 6.2:* [KAOS](#) agents are mapped into [SoaML](#) participants.

As a SoaML serviceContract is the specification of service as to what information, products, assets, value, and obligations will flow between providers and consumers, then the candidate service identified in the KAOS model can be directly mapped to the SoaML serviceContract.

A SoaML participant represents something that provides and/or consumes services, and a KAOS agent is who provides or consumes a system goal (e.g., requirement implemented as a service). Thus, SoaML participant can be mapped in KAOS agent.

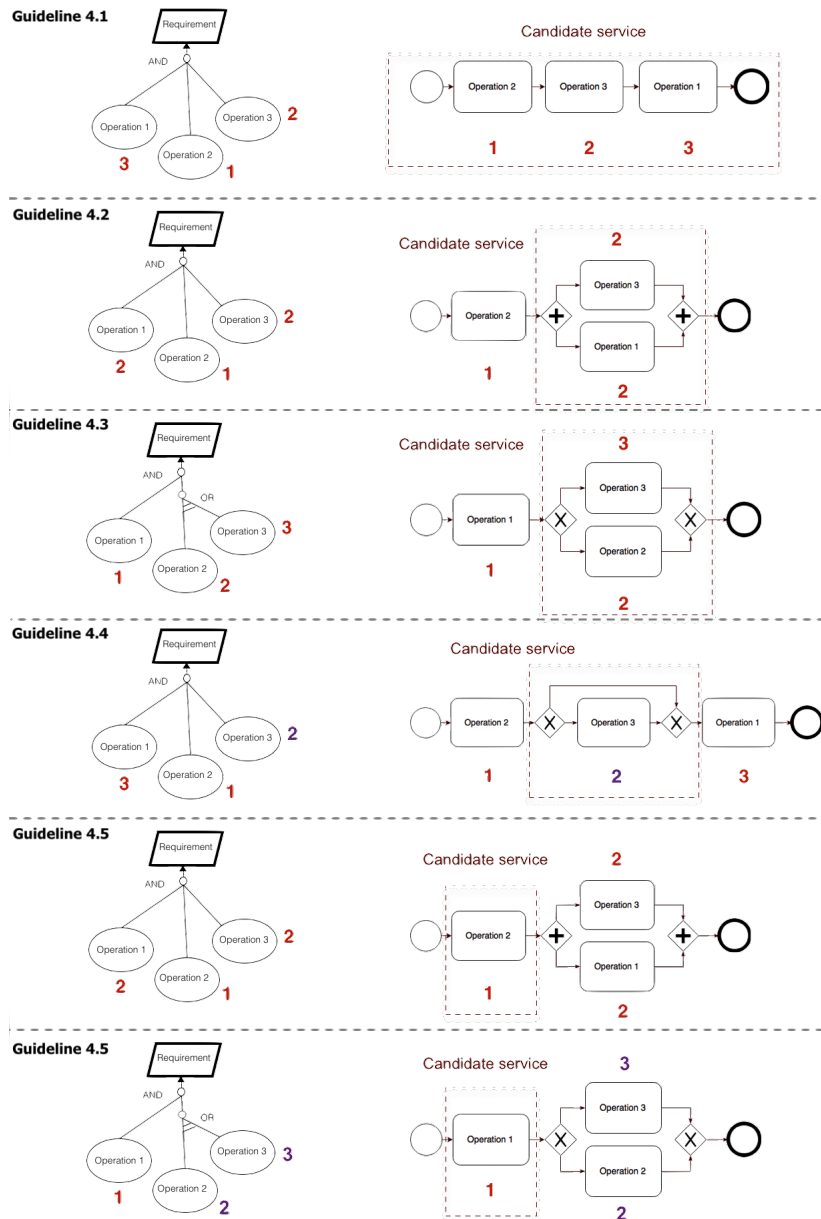


Figure 5.23: Candidate services, using guidelines 4.1–4.5. The colored numbers indicate the order of the operations or activities.

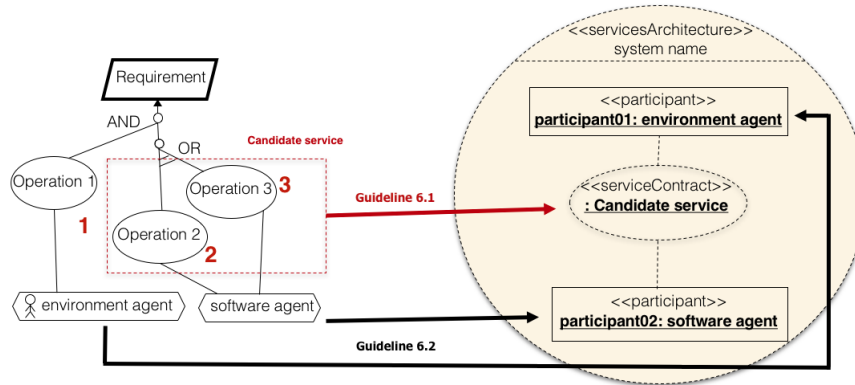


Figure 5.24: Applying guidelines 6.1 and 6.2

5.4.5 About the KAOS4Services implementation

We used the Eclipse Sirius [80] to implement the DSLs for KAOS4Services. The Eclipse Sirius is an open Source framework developed with the purpose of easily and quickly create a graphical modeling workbench dedicated to a domain specific language. It allows developers to graphically design complex systems while keeping the corresponding data consistent. The modeling workbench created with Sirius comprises a set of Eclipse editors that allow developers to create, edit and visualize EMF models.

In comparison with Eclipse EuGENia, we only highlight a weakness that is the non-support of languages like EVL for semantic validation of the created models and meta-models. Instead of EVL, we used AQL [92] that is a language used to navigate and query an EMF model. In general, we felt that using AQL is more complex than using EVL, although it is easily extensible with Java classes.

5.5 Final considerations

This Chapter describes the value-based framework for software architecture. This framework is composed of three core modules: Business Value Modeling, Agile Reference Architecture Modeling, and Goal-Driven SOA Architecture Modeling. It is supported by a set of MDD languages, transformations and tools developed using an Eclipse-based Implementation Environment. While the Business value modeling module focuses on building a stakeholder-centric business specification, the Agile Reference Architecture Modeling and the Goal-Driven SOA Architecture Modeling modules concentrate on generating a reference software architecture aligned with the business value specification described in a DVD model.

The feasibility of this framework is illustrated in the next chapter using a case study. The three methods (DVD, RAMA and KAOS4Services) are used and their application is discussed thoroughly. Later, Chapter 7 discusses a set of controlled experiments and quasi-experiments to check, mainly, the perceived easy to use and usefulness of the methods that make up the proposed framework.

CASE STUDY

To illustrate the use of our Value-Driven Framework we chose a case study inspired in an online auction system that was part of a Brazilian gas station chain fidelity program. This chapter starts with the construction of a value model for that business domain using the [DVD](#) method and follows applying the whole process of deriving software reference architecture models using the [RAMA](#) and [KAOS4Services](#) methods and respective tools.

6.1 Business description

A summary of this system is as follows. When a gas station chain customer registers in the system, he earns 50 coins to bet in any auction of the system (each coin allows one bet). Additional coins are acquired if a customer (i) shops in a gas station (receives the product and coins), (ii) wins an auction (places a bet and expects to be the winner), or buys coins packages (pays for coins). The system provides several auctions concurrently, always selling cheaper than market price. The idea is not to earn by selling a third (partner) company's product or service (i.e. goods) but by having a large number of bets or selling its own goods. An auction starts with a minimum, current and maximum price of goods, a start time, and an envisaged end time. It begins with the minimum price and, each time a bet is placed, the current price is increased by R\$ 0,01. If the auction finalizes before reaching the maximum price, the customer makes a very good acquisition (paying much less than market price). If the price reaches the maximum price, he is still acquiring the good cheaper than in the market. Thirty seconds from the deadline, a new bet postpone the end time in thirty seconds, allowing time for more bets. The winner is the owner of the last bet, who is contacted by e-mail and has thirty days to pay with a credit card for the good acquired. The credit card company must provide a secure financial service in exchange of a payment. After the payment is confirmed, the gas station uses a delivery

service to deliver the product to the customer. If payment is not concluded, the gas station chain creates a new auction with that same product. The online auction system of the gas station chain sells advertisement, receiving goods to be auctioned in exchange for publicity in their own website (large number of visualizations).

The starting point of the Value-Driven Framework is the DVD method to produce a DVD model expressing actors (for the system focus and context), value exchanges, value objects and quality criteria. From the DVD model, we can choose an agile setting and derive conceptual models from where the architectural model is derived or chose a goal-oriented approach that uses a set of guidelines to derive a software reference architecture from a KAOS model.

6.2 Applying the DVD method

We started creating the dvd model by analyzing the problem description following the steps defined in the process depicted in Figure 5.6. Let's use the DVD tool to create the DVD value model using the elements present in the editor palette (Figure 6.1-a). The palette is divided into two sections: Elements, containing the necessary the base DVD value model concepts, and Relations, containing the relationships to associate the model concepts.

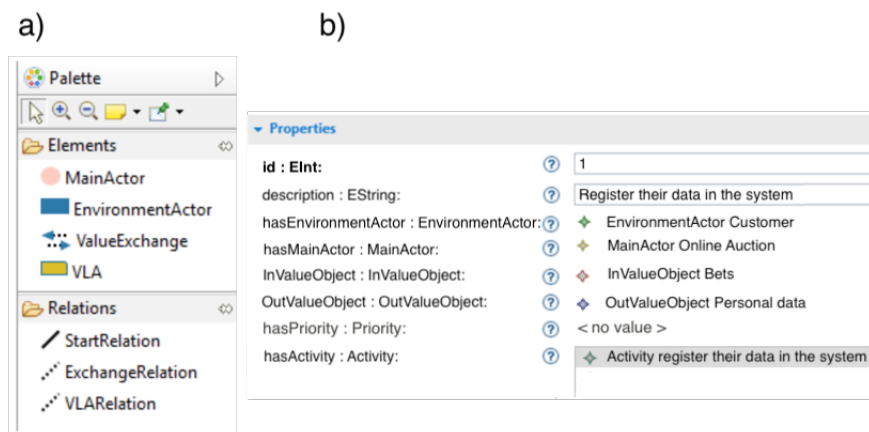


Figure 6.1: (a) DVD editor palette and (b) the value exchange properties.

Identify main actor and environment actors. Analyzing the presented business description, first, we identified *Online auction*, *Customer*, *Partner*, *Credit card company*, and *Delivery company* as the actors of this business (see Figure 6.2). *Online auction* is the focus of the analysis (main actor) and the other actors are those with whom value exchanges occur (environment actors). The link between the environment actors and the main actor is made through the relation *StartRelation*.

Identify value exchanges and who starts each value exchange. The four value exchanges with *Customer* are (i) register their data in the system, (ii) buy in the gas station, (iii) place bidding, (iv) pay for product won at auction. In these value exchanges, customers start the actions. The *Partner'* offers goods to be auctioned and start the action. The *Credit card* company offers a financial service contracted by the *Online auction* that starts the action. The *Delivery company* offers the delivery service to deliver goods to the winners, and the *Online auction* starts the action. It is important to note that each value exchange is graphically represented in the model through the ValueExchange element and is identified by an ID and a textual description (see Figure 6.1-b).

Identify value level agreement. The VLAs for the value exchanges with *Customer* are *free coins* and *low cost* of good, with *Partner* is *large number of visualizations*, with *Credit card company* is *Security*, and with *Delivery company* is *fast* (see Figure 6.2). To facilitate the illustrative example, we created one activity for each value exchange (with the same value exchange's name, as can be seen in *Activity* property in Figure 6.2-b).

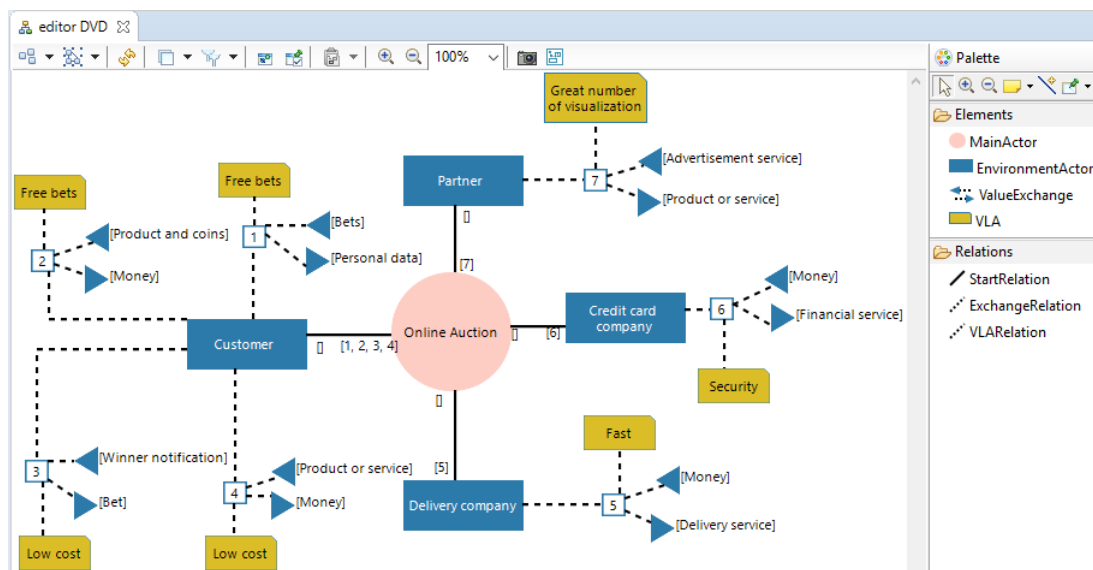


Figure 6.2: DVD model created using the tool.

Change the main actor. This activity allows the analyst to change the focus of his analysis to another actor. Although we feel that it is not necessary to change the focus of the analysis in this case study, we will illustrate what this change would look like. Let's choose "Customer" as the focus of our analysis. Figure 6.3 shows the corresponding DVD model, where "Customer" becomes the main actor, and the previous main actor "Online Auction" becomes an environmental actor. Now, the analyst can start analyzing the value exchanges from the "Customer" point of view. Also, the analyst does not lose what was previously specified for the actor "Online Auction".

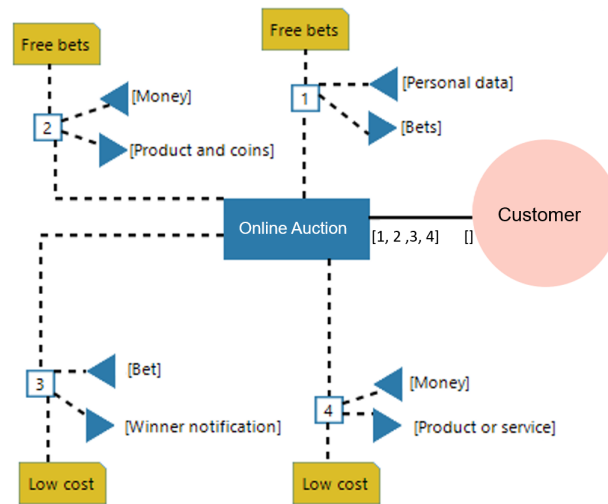


Figure 6.3: Changing the focus of analysis.

Generating SoaML capability model. Once created the DVD value model of the virtual store, we can opt to create a SoaML capability model, or we can start the architectural derivation process using the RAMA or KAOS4Services methods. For example, we may want to apply a transformation script, such as the one described in Listing 5.2, to generate a SoaML capability model. To do this, we open the execution settings (see Figure 6.4) to create a transformation and define the name of the transformation.

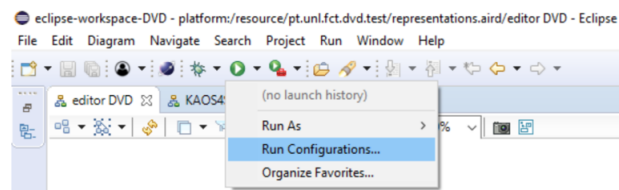


Figure 6.4: Menu to open the transformation execution window.

Next, we choose the ETL file corresponding to the transformation of the DVD model to the SoaML capability model (DVD2Capability.etl), as shown in Figure 6.5.

To conclude, we perform the transformation to generate the SoaML capability model in Figure 6.6. Note that we have created an abstract element (“General capability”) to aggregate all the capabilities generated from the value exchanges. In other words, this abstract element is the parent of the capabilities generated from the activities.

6.3 Applying RAMA method

Similarly to the application of the DVD method, we follow closely the RAMA method process summarized in Figure 5.17.

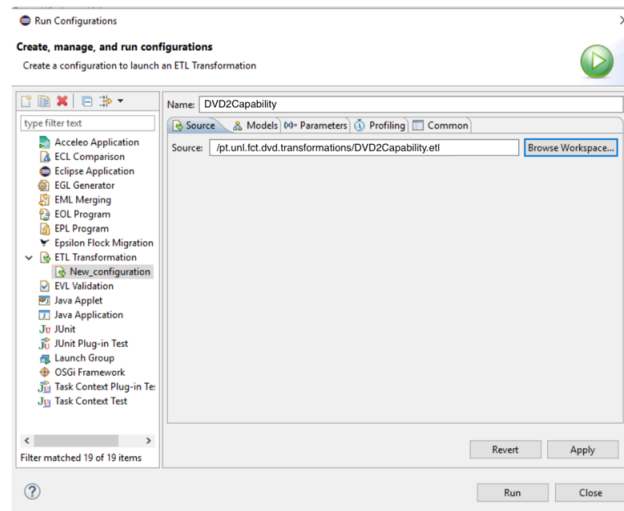


Figure 6.5: Execution window.

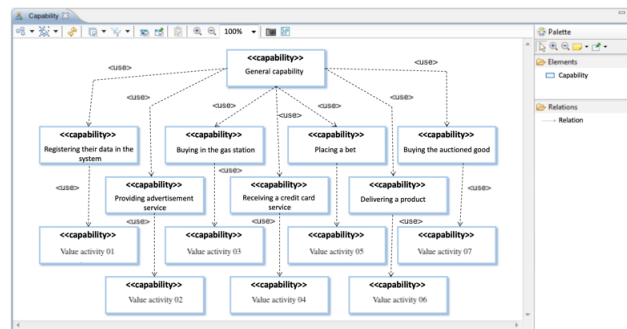


Figure 6.6: The capability model generated.

Specify user stories. We specified the user stories necessary to satisfy each value exchange. Each user story is then described using the editor we created based on the BehaviorMap tool. Table 6.1 describes all user stories specified.

Table 6.1: User stories

ID	Value Exchange	Actor	User story
VE1	Register their data in the system (Registering data)	As a customer	GIVEN I do not have an account WHEN I create an account THEN I see the active auctions.
		As a customer	GIVEN I am creating an account WHEN I finish registering THEN I get bets (coins) free.
VE2	Buying in the gas station	As a customer	GIVEN I have an account WHEN I buy goods (products) in the gas station

Continues on next page

Table 6.1 – Continuation from previous page

ID	Activity	Actor	User story
VE3	Placing a bet	As a customer	THEN I earn free bets (coins).
			GIVEN I'm logged in
			WHEN I see all auctions available
			THEN I can choose where I will place my bets.
		As a customer	GIVEN I'm logged in
			WHEN I place a bet on an auction
			THEN I can be the winner.
		As a customer	GIVEN I'm participating in an auction
			WHEN I want to know if I am losing an auction where I am betting
			THEN I see who is the current winner.
		As a customer	GIVEN I'm logged in
			WHEN the auction is available
			THEN I can make another bet if I am not winning the auction.
		As a customer	GIVEN I'm participating in an auction
			WHEN I am the winner
			THEN I want to be notified by email.
			GIVEN I'm participating in an auction
			WHEN I am the winner
VE4	Buying the auctioned good	As a customer	THEN I want to know the price of the good I won so that I can pay for it.
			GIVEN I'm participating in an auction
		As a customer	WHEN I am the winner
			THEN I want to know the deadline to confirm the payment so that I can perform the payment on time.
			GIVEN that I have an interest in advertising a product or service
			WHEN I provide a product or service to be auctioned
VE5	Providing an advertisement service		THEN I want to see my ad during auctions with products/services of my interest.
			GIVEN that the online auction wants to use my payment service by credit card
			WHEN I get paid for the service
			THEN the service will be available through an Application Programming Interface (API) respecting the agreed deadlines.
			GIVEN that I provide the products of the auctions
			WHEN I get paid for delivery
VE6	Receiving a Credit card service	As a credit card company	THEN I will deliver the product in accordance with the pre-established deadlines.
VE7	Delivering a product	As a Delivery company	

Continues on next page

Table 6.1 – Continuation from previous page

ID	Activity	Actor	User story
----	----------	-------	------------

Figure 6.7 illustrates the specification of the user story to create an account, that is, the first user story in Table 6.1.

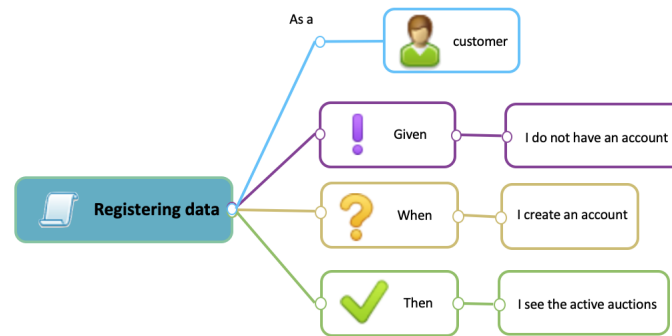


Figure 6.7: “Registering data” user story.

Prioritize value exchanges. The prioritization of value exchanges are done based on their **RoI**. This prioritization is used to guide the development order of the various activities (e.g., the value exchanges with higher priority will be implemented first). Thus, the business person sets the customers’ value exchanges priority to “high”, the financial value exchange to “medium”, and the remaining value exchanges as “low”. Figure 6.8 shows the prioritization action where the user defines the value exchange VE7 with “low” priority. In this case study, we defined an iteration with only the most important value exchanges, that is, those with high prioritization, and we illustrate the development of this iteration only, as the others are performed in a similar way. Therefore, we execute the second prioritization round only for the value exchanges VE1–VE4.

Figure 6.9 shows the tree model with the result of the second round of prioritization, for VE1–VE4. This tree model was generated similarly to the capability model, that is, we open the execution settings option in the eclipse platform, as shown in Figure 6.4 and choose and perform the **ETL** file corresponding to the generation of the tree model (see Figure 6.9). It is important to highlight here that the implementation of this tree model generation is as proof of concept. For an end-user tool, this model must be generated and updated automatically, i.e., all data should be stored in a database, and some Observer pattern¹ should be implemented to always keep the tree model updated according to the change of data. Another essential point to be highlighted is that the hierarchical structure of the model facilitates concept traceability. For example, we can identify all user stories associated with a value exchange visually because a value exchange is shown as a parent of the user stories.

¹The Observer pattern defines a relationship among objects so that when one object changes state, the others are notified and updated automatically [98].

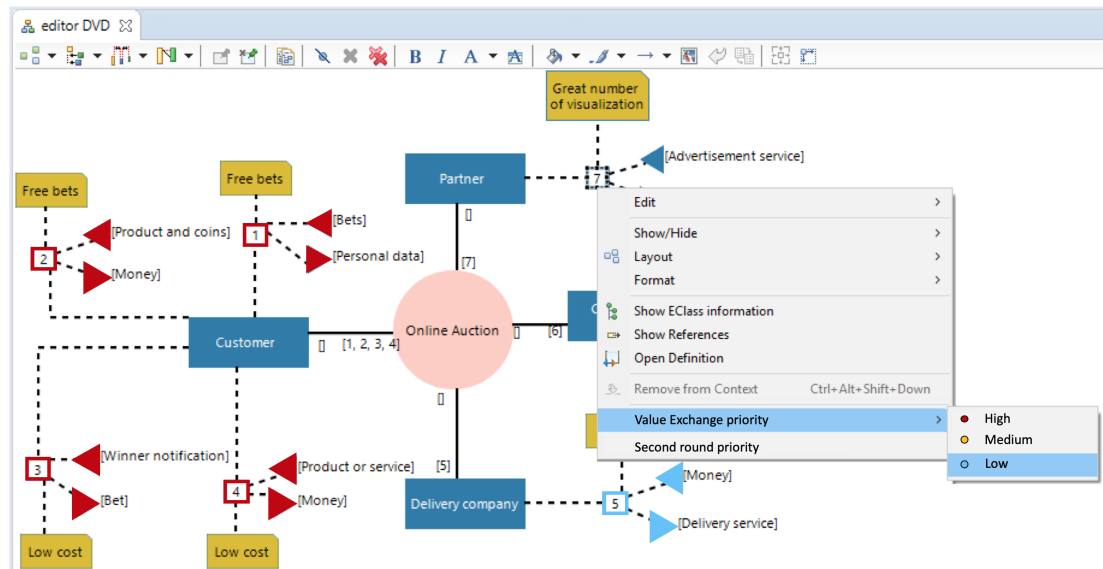
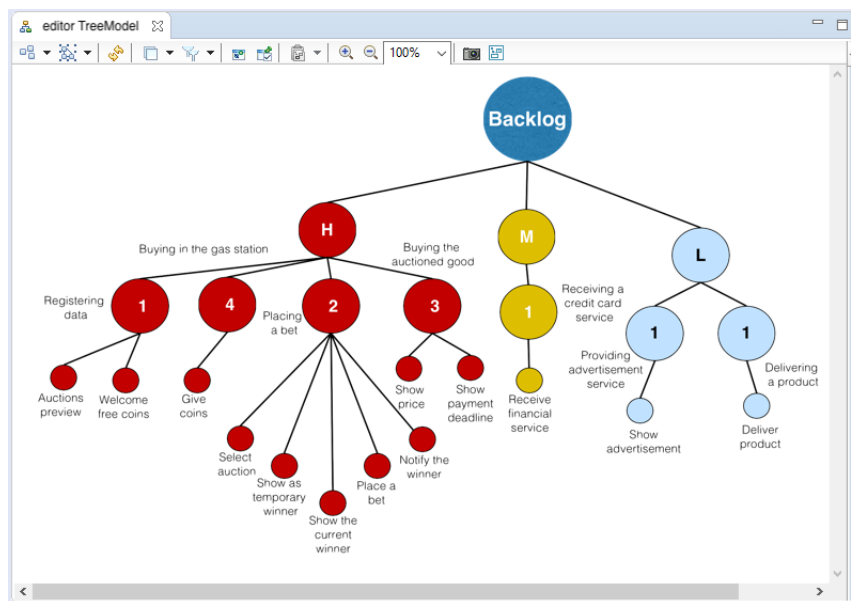


Figure 6.8: Prioritization of value exchanges.

Figure 6.9: Tree model generated after prioritization. Note that the value exchanges are represented grouped in **H**igh (red), **M**edium (yellow), and **L**ow (blue) priorities according to RoI.

Specify conceptual models. We used the SimpleMind Lite tool [178] to create conceptual models specifications structured as mindmaps. For example, Figure 6.10 illustrates a conceptual model for VE1. The central node is the name of the value exchange “Registering data”. We identified that a Customer and a Partner must have accounts. Also, Customer must save some mandatory data, for example, personal data, billing data, delivery data. In addition, each Customer must have a wallet and this wallet knows the customer’s quantity of coins and holds a transactions history.



Figure 6.10: Conceptual model for “Registering data” user story.

The central node of VE2 is “Buying in the gas station” (see Figure 6.11).

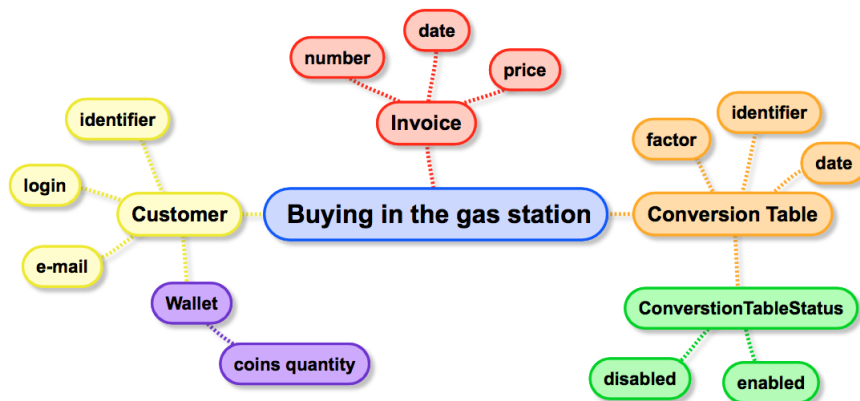


Figure 6.11: “Buying in the gas station” conceptual model.

When a customer buys a good in the gas station, an invoice is issued. Later, Customer must inform the system about the data of the invoice so that it calculates (conversion table) the number of coins to be added to the customer’s wallet. For VE3, the central node is “Placing a bet” (Figure 6.12). Customer places a bet in an auction. The auction offers an object (service or product) and saves the history of all the bets. When the auction finishes, a notification is sent to the winning customer.

For VE4, the central node is “Buying the auctioned good” (Figure 6.13). Customer knows the price of the auction s/he won and offers billing data to complete the payment process. During this process, a monitor checks that all payment steps (e.g., if the payment was performed before the expiry date) and all changes that may happen during the process are registered (history). When payment is concluded and the object auctioned is of type

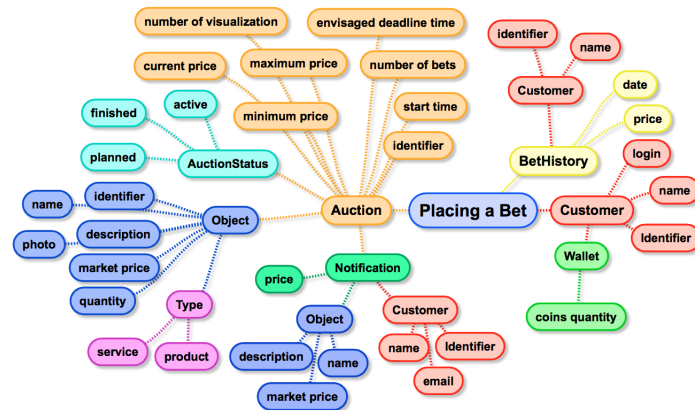


Figure 6.12: Conceptual model for “Placing a bet” user story.

product, the delivery process starts.

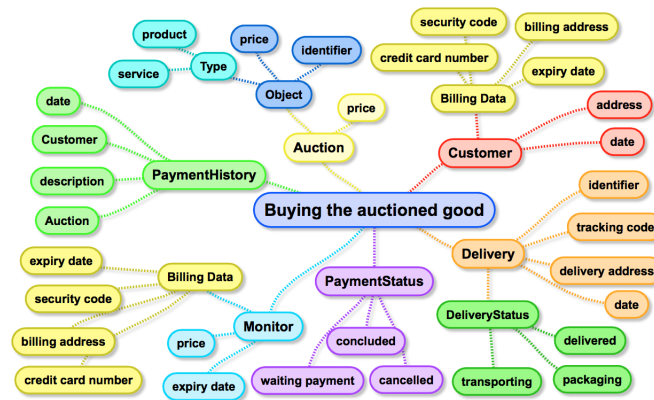


Figure 6.13: Conceptual model for “Buying the auctioned good” user story.

Identify concepts overlaps & decision analysis. Using Levenshtein algorithm [164], a total of eight overlaps were found. Table 6.2 shows the overlaps and the decisions made. This algorithm sufficient for a proof of concept tool, but for a fully fledged tool, the algorithm should at least identify synonymous words. For example, in the first overlap identification, if we had defined the customer with the word “client” or “consumer” or “purchaser”, no overlap would have been found, even if these concepts were conceptually the same.

Generate reference architecture. To finalize the process of combining architecture design and agile practices, the development team automatically generates the reference architecture model, using the transformation script in Listing 5.7, and names each component (AP09). For that, they must open the execution settings option in the eclipse platform and choose and perform the ETL file corresponding to the generation of the architectural model. After the architectural model is generated, we open it using the

Table 6.2: Overlaps

#	Conceptual model A	Conceptual model B	Overlap concept	Decision analysis
1	Registering data	Buying in the gas station	Customer	Create new “Customer”
2	Registering data	Placing a bet	Auction	Create new “Auction”
3	Registering data	Placing a bet	Object	Create new “Object”
4	Buying in the gas station	“Customer”	Customer	“Customer”
5	Placing a bet	“Customer”	Customer	“Customer”
6	Placing a bet	Buying the auctioned good	Auction	“Auction”
7	“Auction”	“Object”	Object	“Object”
8	Buying the auctioned good	“Customer”	Customer	“Customer”

Obeo UML Designer tool [186]. Figure 6.14 shows the reference architecture model generated where the “registering data” was renamed to “Partner”, “placing a bet” to “Bet”, and “Buying the auctioned good” to “Payment”. Component relationships are detected automatically through the concepts overlaps, creating component interfaces (offered and required) with the name of the concept. The offered interfaces are represented by a lollipop-symbol and a required interface by an open socket symbol.

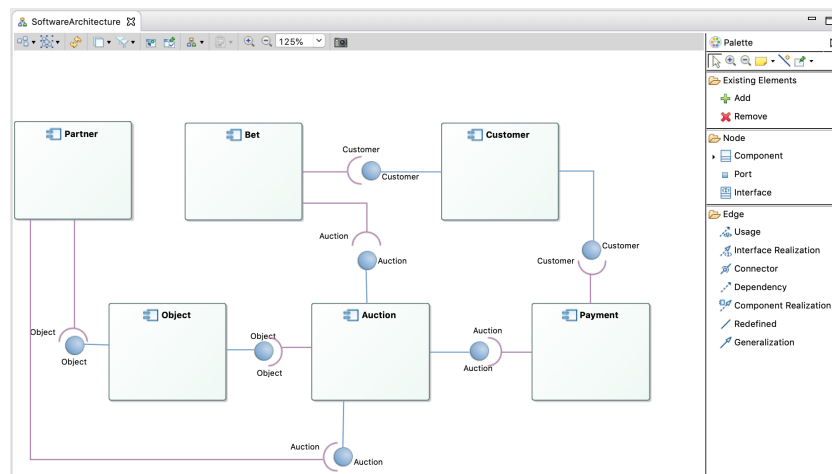


Figure 6.14: Reference architecture for the online auction system.

For example, the *Customer* component offers the concept “Customer” to the components *Bet* and *Payment*. The interface “Customer” between the components *Customer* and *Bet* is used to register the customer who places the bet; and the interface “Customer” between the components *Customer* and *Payment* is used to register the customer who makes the payment. *Auction* component offers the concept “Auction” to the *Bet* component to add a bet in an auction to *Partner* to register the auctions that the partner sponsors and to *Payment* component to complete the auction. Finally, the component *Object* offers the concept “Object” to the component *Auction* to register the object that is being auctioned and to the component *Partner* to register the object offered to be auctioned by the partner.

Thus, this reference architecture encompasses the necessary knowledge on how to design a concrete architecture for an online auction system, aligned with the value exchanges of an online auction.

6.4 Applying KAOS4Services method

Similarly to the previous sections, the discussion of illustration of the **KAOS4Services** method will also follow the structure of the corresponding process depicted in Figure 5.21.

Generate a KAOS model. A **KAOS** model is generated from a **DVD** model by applying the mapping described in Section 5.4.4 and using only the high-priority value exchanges. This is achieved by opening the execution settings option in the eclipse platform, choosing and executing the **ETL** file. Figure 6.15 shows the initial **KAOS** model for the case study, starting with the abstract goal root “Implement auction online system”, which is decomposed into subgoals the “Registering data”, “Placing a bet”, “Buying the auctioned good”, and “Buying in the gas station”.

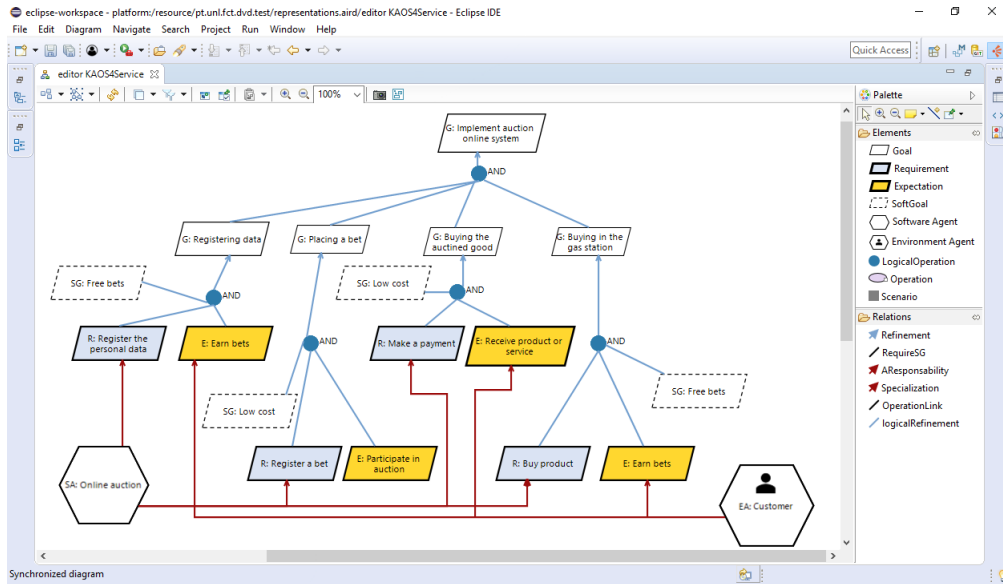


Figure 6.15: Initial KAOS model for the auction online system.

To satisfy “Registering data” we need to achieve “Free bets” (softgoal), “Register the personal data” (requirements), and “Earn bets” (expectation). To satisfy “Placing a bet”, we need satisfy a softgoal “Low cost”, requirement “Register a bet” and expectation “Participate in auction”. To satisfy “Buying the auctioned good”, we need to achieve “Low cost” softgoal, “Make a payment” requirement and “Receive product or service” expectation. Finally, to satisfy “Buying in the gas station”, we need satisfy at least softgoal “Free bets”, requirements “Buy product”, and expectation “Earn bets”.

Note that by clicking on one of the generated elements, the information of its origin is displayed in its properties. For example, Figure 6.16 shows the property tab of the “Placing a bet” element, showing that its source comes from the Value Exchange element with ID 3 of the **DVD** model. With this information we confirm the traceability between

the elements of the models and to ensure that this traceability remains throughout the development process.

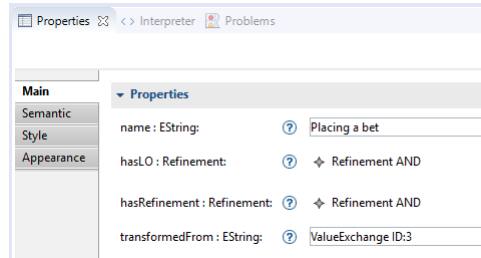


Figure 6.16: Properties of “Placing a bet” requirement.

Decompose requirements. This activity focuses on the requirements (sub)goals to specify behaviors the system needs to achieve. Note that expectations are human-intensive operations and we are only interested in system-intensive operations, although there is some system-intensive operation to satisfy a user expectation. Next we decompose “Register the personal data”, “Register a bet”, “Make a payment”, and “Buy product”.

Register the personal data. Figure 6.17 shows the decomposition of “Register the personal data”. For “User registration”, a customer must provide his data and submit them to the system, then the system validates the customer’s data and notifies data error in case of an error. In contrast, the system must save the customer’s data and give 50 coins to the customer. For “User authentication”, customer submits his login and password and the system validates its authentication. If an error occurs, the customer is notified (conditional operation, so we use purple). The order of the operations in the model helps establishing a behavioural scenario.

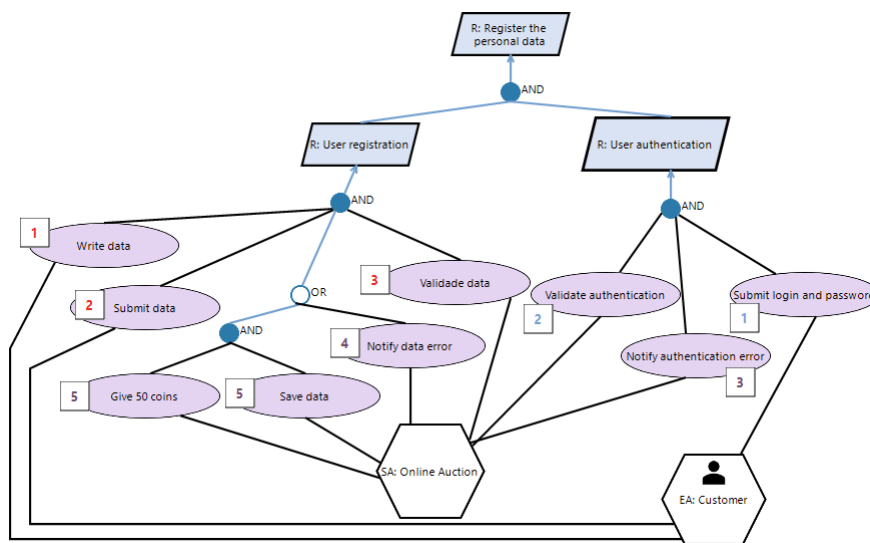


Figure 6.17: Decomposition of requirement “Register the personal data”.

Register a bet. Figure 6.18 decomposes “Register a bet” into requirements “Update

action data” and “Validate bet”. “Update action data” updates the auction data (e.g. current winner, auction time, price) after a bet, and “validate bet” checks some business rules (e.g., verify coins in customers wallet, check if the auction time is over, and notify the customer when these rules are not confirmed). If the user has a coin in his wallet and the auction time has not ended, the system must update the auction data informing about the current winner, increase the auction time and price, decrease one coin of the customer’s wallet, and update the bet historical data.

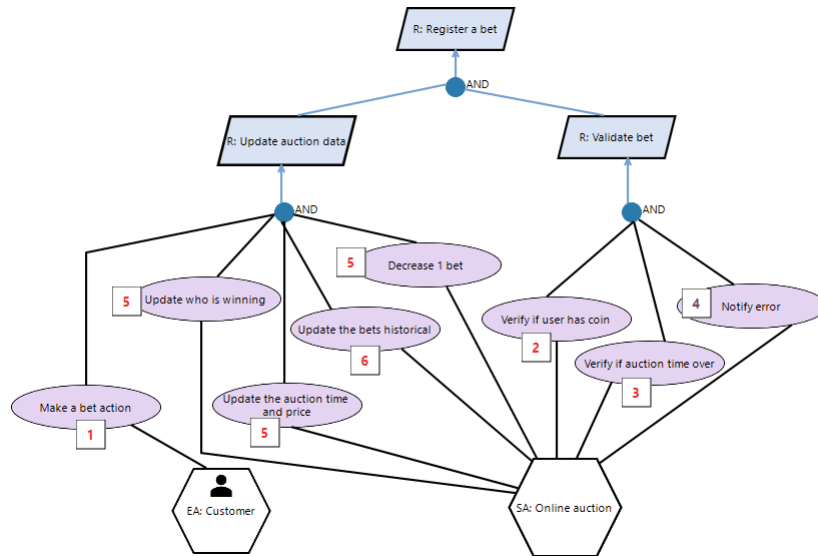


Figure 6.18: Decomposition of requirement “Register a bet”.

Make a payment. Figure 6.19 shows the decomposition for “Make a payment”. As the process is similar to the two previous ones, and to avoid repetitions, we show only the “Make checkout” decomposition.

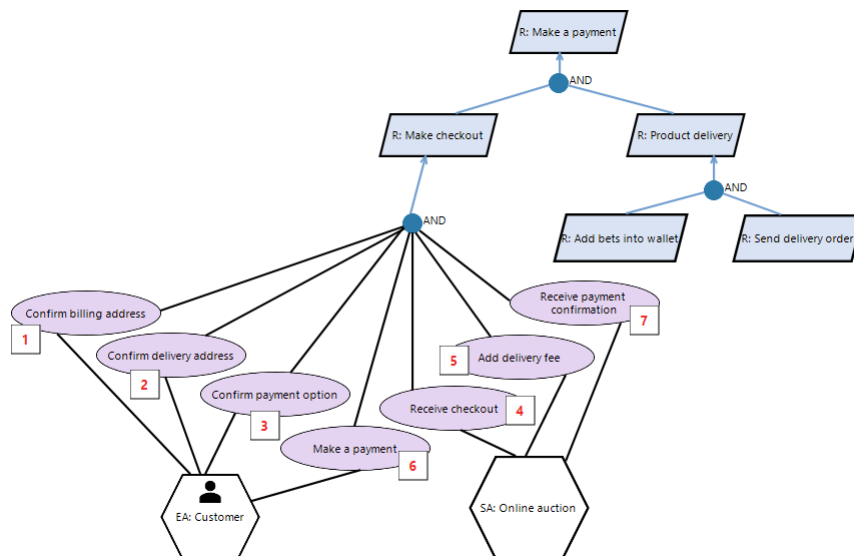


Figure 6.19: Decomposition of requirement “Make a payment”.

To checkout, the customer confirms the payment address, delivery address and payment option. The system then receives the checkout and adds the product delivery rates. Once the delivery rates are calculated, the customer must make the payment, causing the system to receive confirmation of this action.

Buy product. Figure 6.20 shows the decomposition for “Buy product”. The customer “informs the invoice data” so that the online auction system “calculates the number of bets” the customer will receive. The number of bets is in conformance to the amount of the invoice. Once the calculation is performed, the system must “identify the customer” and “add the number of bets in customer’s wallet”.

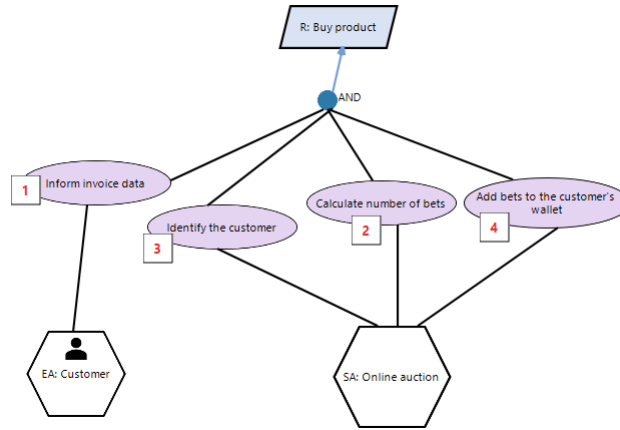


Figure 6.20: Decomposition of requirement “Buy product”.

Generate business process. Figures 6.21 to 6.25 show the business process models generated from the KAOS models using model-driven techniques by applying guidelines 3.1 – 3.5. The guideline 3.1 applied to “user registration”, transforms its agents “customer” (bidder) and “online auction” to pools, the guideline 3.2 transforms its operations into BPMN activities (with the same name) in the pool of the same agent, and the guideline 3.3 keeps their order. Guideline 3.5 takes the conditional behaviour represented by the OR operator and inserts the affected operations in the BPMN model using the exclusive operator. Note that guideline 3.4 takes the operations with the same order (number 5) and inserts them in BPMN parallel gateways by applying the guideline 3.4. And similarly to all the operations in the rest of the KAOS models.

Agents and operations were transformed into BPMN pools and activities, respectively. The order of the operations is kept for the activities and the logical AND and OR operators from KAOS were also considered during the model transformation.

Update business process. This activity, strongly dependent on the expertise of the designer, is only needed if a more detailed specification is required. As the level of abstraction of the obtained BPMN processes is adequate, this step was not performed.

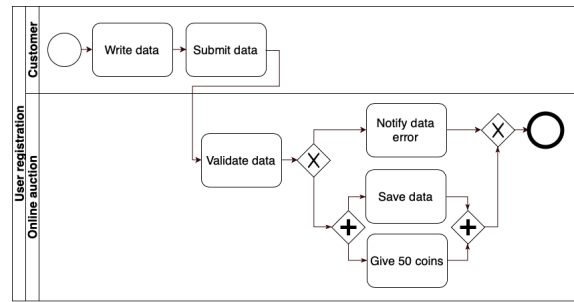


Figure 6.21: Business processes generated from the requirement “user registration”.

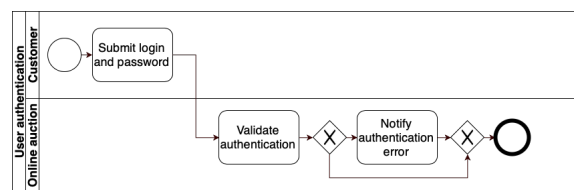


Figure 6.22: Business processes generated from the requirement “user authentication”.

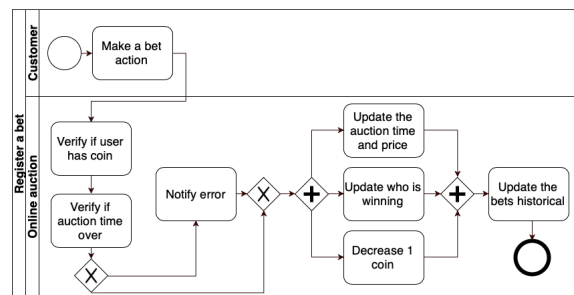


Figure 6.23: Business processes generated from the requirement “register a bet”.

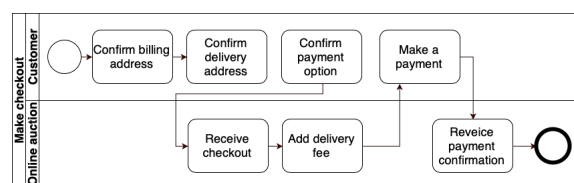


Figure 6.24: Business processes generated from the requirement “make checkout”.

Identify candidate services. The user selects the system-intensive operations and identifies candidate services using guidelines 4.1 – 4.5. Figure 6.26 shows the “Identify tab” of the graphical interface of the tool to determine the candidate services. First, we enter the path where the KAOS model is saved. Then we press the “Identify candidate services” button and the candidate services are listed in a table, showing their identifiers (e.g., ID), the heuristic used, the origin requirement, and the set of composing operationalizations.

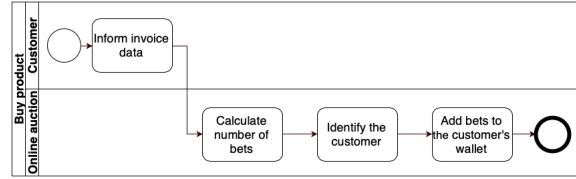


Figure 6.25: Business processes generated from the requirement “buy product”.

Table 6.3 shows lists all the candidate services identified.

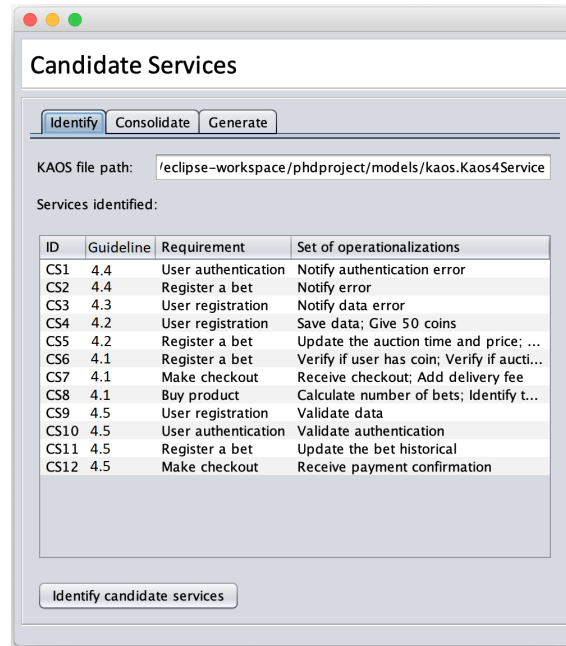


Figure 6.26: Identifying the candidate services.

Table 6.3: Applying guidelines 4.1 – 4.5 to identify candidate services.

#	Guideline	Requirement	Set of operationalizations
CS1	4.4	User authentication	Notify authentication error
CS2	4.4	Make bet	Notify error
CS3	4.3	User registration	Notify data error
CS4	4.2	User registration	Save data, and give 50 coins
CS5	4.2	Make bet	Update the auction time and price, update who is winning, and decrease 1 coin
CS6	4.1	Make bet	Verify if user has coin and verify if auction time is over
CS7	4.1	Make checkout	Receive checkout and add delivery fee
CS8	4.1	Buy product	Calculate number of bets, identify the customer, and add bets to the customer's wallet
CS9	4.5	User registration	Validate data
CS10	4.5	User authentication	Validate authentication
CS11	4.5	Make bet	Update betting historical
CS12	4.5	Make checkout	Receive payment confirmation

For example, for “User registration”, the guideline 4.2 identified the Candidate Service (CS) containing “Save data” and “Give 50 coins” operations (CS4), the guideline 4.3 identified CS3, and the guideline 4.5 identified CS9.

Consolidate candidate services. Guidelines 5.1 – 5.3 are used to help designers decide about which candidate services should be implemented. In this particular case, the guideline 5.1 (reuse) applied to all candidate services results in the same number of goals (four). Therefore, this guideline was not very useful in this case study. For 5.2 (dependency), CS8 stands out from all the others because it has three dependent candidate services (e.g., “verify if auction time over” from CS6, “notify error” from CS2 and “update the bets historical” from CS11 are connected to the operations from CS5). Regarding guideline 5.3 (aggregation), the designer analyzes the candidate services with only one operation. This analysis is performed subjectively. For example, CS3 and CS9, CS1 and CS10, and CS6 and CS2 could form three new aggregations due to their simplicity and size. Figure 6.27 shows the “Consolidate tab” of the tool.

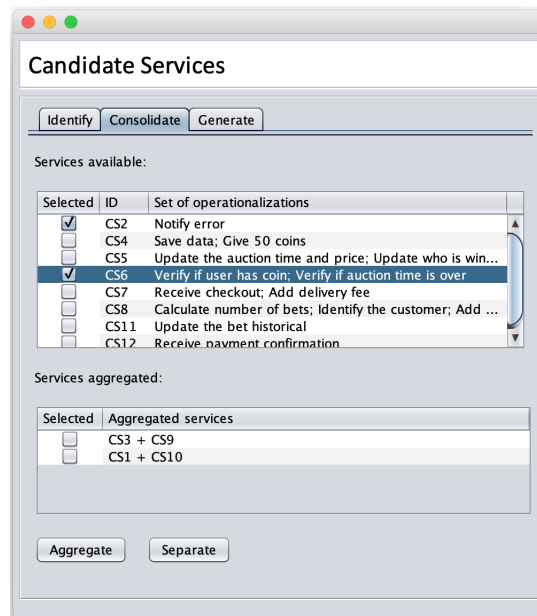


Figure 6.27: Consolidating the candidate services.

To aggregate the candidate services we selected two available services (top table) and pressed the “Aggregate” button. After, we verified if the services were aggregated successfully by checking if they were shown in the table “service aggregated” (bottom table). For example, Figure 6.27 shows our intention of aggregating services CS6 with CS2. One can also see that we aggregated before services CS3 with CS9 and CS1 with CS10. If we wanted to separate the candidate services that were aggregated then we just needed to select the aggregated service in the lower table and then press the “Separate” button.

Generate service specification. This activity automatically generates a reference architecture for software services using guidelines 6.1 and 6.2, mapping **KAOS** concepts into **SoaML** concepts through model transformations. Guideline 6.1 transforms “candidate services” into **SoaML** “serviceContract”, and Guideline 6.2 transforms **KAOS** “agents”

into SoaML “participants”. Figure 6.28 shows the “Consolidate tab” of the tool. Note that the user can select which candidate services he wishes to model. In our case, we selected all candidate services and pressed the “Generate SA model” button.

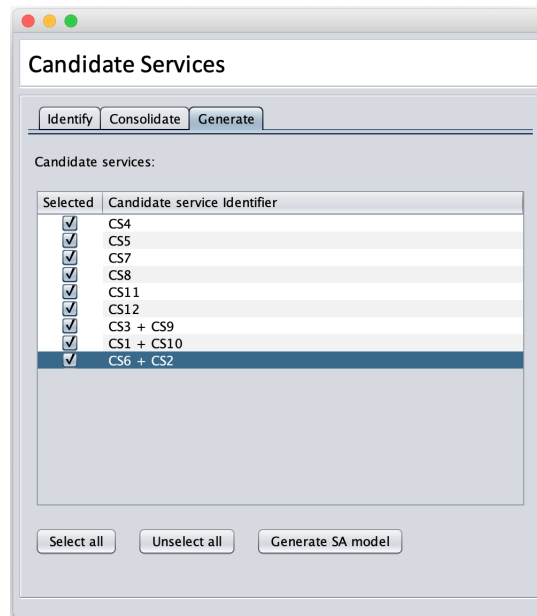


Figure 6.28: Generating the architectural model.

Figure 6.29 shows the resulting SoaML model, where the designer is expected to name each service. This service specification offers a realistic idea of what must be implemented in the next software development phases.

6.5 Final considerations

This Chapter uses an online auction system that is part of a Brazilian gas station chain fidelity program to illustrate the use of our value-framework to derive a software reference architecture. It starts with the construction of a value model with the DVD method and proceeds using RAMA and KAOS4Services for two alternative ways of deriving a software reference architecture model. During the execution of the case study, we used our proof-of-concept tools and verified the traceability of the concepts during the transformation tasks and the feasibility of the three methods (DVD, RAMA, and KAOS4Services).

Note that we are able to trace concepts between models using model transformations and metamodeling approaches. However, to trace DVD value concepts to the architectural model, it is required to keep track of the concepts within the models themselves, not just between the models. To achieve this, we were careful to use only hierarchical models, to be able to identify the parents’ and children’ concepts. For example, a model structured as a mindmap (e.g., DVD model) has internal traceability from its central node to its border nodes. Similarly, the KAOS model used by KAOS4Services is a hierarchical model, where a more abstract goal is decomposed into more refined goals recursively

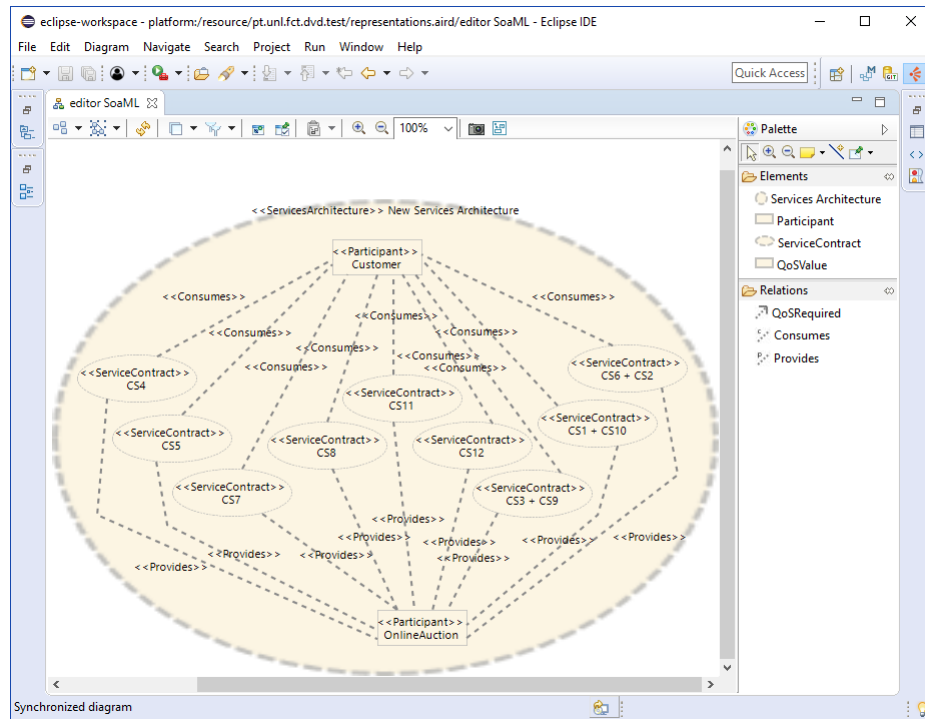


Figure 6.29: SoaML model for online auction system.

until operations are reached. For example, a DVD value exchange is transformed into a KAOS high-level goal. Next, this goal is decomposed into (sub)goals down to operations. So we can track which KAOS model operations are related to the high-level goal that is linked with the DVD value exchange concept. Also, as an architectural candidate service is composed by a set of KAOS operations, we can trace such service to a value exchange using backward traceability.

While this chapter illustrates the application of the methods offered by the Value-Driven Framework, the next Chapter shows the planning and execution of a set of experiments performed to evaluate the perceived ease of use, perceived usefulness and intention to use by the one hundred and thirty one participants involved in the experiments. We also compare DVD with e3value as well as RAMA with KAOS4Services.

EVALUATION THROUGH EXPERIMENTS

There are four relevant methods for conducting experiments in the area of software engineering [265]: scientific, engineering, analytical, and experimental.

The scientific method uses an inductive paradigm and tries to extract from the world some model that can explain a phenomenon and to evaluate if the model is representative of the phenomenon under observation. The engineering method uses an evolutionary improvement-oriented approach that looks at existing solutions and suggests the most appropriate one. The analytical (or mathematical) method uses a deductive approach that suggests a formal theory to evaluate solutions. Finally, the experimental method uses a revolutionary improvement approach to create or to evaluate a solution. For this, it develops the qualitative and quantitative method, applies an experiment, measures, analyzes, and evaluates hypotheses about a solution, and repeats all the process until reach the answers to the hypotheses. The process begins with the survey of a new solution (not necessarily based on an existing solution) and studies the effect of the process or product suggested by the new solution. The experimental method is considered the most appropriate approach to experimentation in the area of Software Engineering [254, 265].

This chapter discusses a set of different experiments used to evaluate our proposal. To reduce the risk of bias during the evaluation of our proposal, we performed at least one experiment per method (DVD, RAMA, and KAOS4Services), hence evaluating the whole proposal. First, we evaluated the *perceived ease of use and perceived usefulness* of the DVD method. With positive results in this experiment, we conducted a family of three experiments controlled to compare our DVD method with the well-known e3value (introduced in Chapter 2.3.2) with respect to their effectiveness, efficiency, perceived ease of use, perceived usefulness and intention to use, completing the evaluation of the framework's business layer.

For the framework's software layer, we evaluated the *perceived ease of use and the*

perceived usefulness, of the [RAMA](#) and [KAOS4Services](#) methods. We also evaluated the participants' intention to use the methods in the future (if necessary). Finally, we compared the results obtained for these two methods, to understand which one approach (agility versus use of goals) was better accepted.

7.1 Evaluating the DVD method

A controlled experiment was performed to evaluate the perceived efficacy (that is, the *perceived ease of use* and the *perceived usefulness*) of the [DVD](#) method. Thus, the research questions for our evaluation are as follows:

RQ1 : *Is the DVD method perceived as easy to use?*

RQ2 : *Is the DVD method perceived as useful?*

7.1.1 Experiment design

Following the [Goal Question Metrics \(GQM\)](#) process [22], the goal of this experiment is to **analyze** the method to create [DVD](#) model **for the purpose of** verifying the perceived efficacy of the method **with respect to** the perceived ease of use and the perceived usefulness, **from the point of view of** novice business analysts and software engineers, **in the context of** postgraduate students in Computer Science.

Context of the experiment: We focused our evaluation on novice modelers since one of our goals is to provide an easy and useful language that will help less experienced modelers to specify business values. Although designed for business analysts, [DVD](#) can be used by business analysts or software engineers. The participants of the experiment are 37 [Master of Science \(MSc\)](#) students in Computer Science at the [Universidade Nova de Lisboa \(UNL\)](#), attending the “Software Engineering” and the “Requirements and Software Architecture” courses. The students were asked to accomplish this experiment as a part of a series of optional lab exercises of the courses. All the students were volunteers and were aware of the practical and pedagogical purposes of the experiment, but they did not know the experimental hypotheses nor did they had previous knowledge on [DVD](#).

Hypotheses formulation: We formulated two null hypotheses, defined in a one-tailed manner, as we want to analyze the effect of the use of our method on the variables. Each null hypothesis and its alternative are as follows:

H1-0: There is no significant perceived ease of use of the [DVD](#) method

H1-a: The [DVD](#) method is perceived as ease of use.

H2-0: There is no significant perceived usefulness of the [DVD](#) method

H2-a: The [DVD](#) method is perceived as useful.

Selected variables and experimental objects: The independent variable of interest is the use of our method with nominal values. Hence, the experiment uses only one treatment: the creation of a DVD model for two business descriptions. The dependent variables are perception-based, assessing the participants' perceptions of their performance after modeling with DVD, hence used to evaluate the perceived efficacy of the language. They are based on **Technology Acceptance Model (TAM)** [66], a widely applied theoretical model to analyze user acceptance and usage behavior of emerging information technologies. It has empirical support through validations and replications [147]. The perceived efficacy [66] of the method can be broken down into two subjective dependent variables:

- **Perceived Ease Of Use (PEOU):** refers to the degree to which a person believes that learning and using our method would not require significant effort.
- **Perceived Usefulness (PU):** refers to the degree to which a person believes that using our method will increase her/his job performance within an organizational context.

These two subjective variables were measured using a 5-point Likert scale questionnaire with a set of 10 closed-questions: 4 questions for PEOU and 6 for PU¹. They were formulated using the opposing statement format (the order of the items was randomized to avoid monotonous responses. This approach avoids result distortions if uncommitted participants always answer 5 - extremely positive - or 1 - extremely negative - on all questions to finish the experiment execution quickly and without carefully reading the questions). So, each question contains two contradictory statements representing the max and min possible values (5 and 1), where 3 is considered a neutral perception. The aggregated value is the arithmetical mean of the answers to the questions associated with each perception-based variable. We used Cronbach's alpha test [184] to evaluate the reliability of the survey and of each variable. Two experimental objects were selected from the literature: waste management [122], and a wireless access provisioning by a hotel [69].

Experiment design: We established two groups (each using one experimental object), and the participants were randomly assigned to each group. Table 7.1 summarizes the design of the experiment. The comprehension of the business description may also affect the application of the language. We alleviated the influence of this factor by selecting two representative business descriptions of a complexity suitable for application in the 1-hour slots available for the execution of the experiments.

Analysis procedure: We chose statistical tests for their robustness and sensitivity to analyze the data collected. As usual, in all the tests we decided to accept a probability of 5% of committing a Type-I-Error [265], rejecting the null hypothesis when it is true. We tested the normality of the data distribution with the Shapiro-Wilk test [223]. To verify

¹The questionnaire can be found at <https://goo.gl/forms/Di88e7wPr27GtbV82>

Table 7.1: Experiment design

Groups	Part 1	Part 2	Part 3
A	Training on both methods	Object1	Post-experimental questionnaire
B	Training on both methods	Object2	Post-experimental questionnaire

our hypotheses, we used the T-test to one sample when data could assume the normal distribution. However, we applied the Wilcoxon-test when the data could not assume the normal distribution.

7.1.2 Discussion of the quasi-experiment results

The use of multiple items to measure the same construct requires the examination of the questionnaire's reliability. We used Cronbach's alpha [184], and the result for the questionnaire was 0.846. This means that the questionnaire is reliable (Cronbach's alpha is higher than 0.7 [184]). After, we analyzed the descriptive statistics on the two variables, as shown in Table 7.2.

Table 7.2: The descriptive statistics for PEOU and PU

Group	PEOU					
	Min.	Max.	Med.	Mean	Std. Dev.	p-value
A	3	5	4	4.1	0.52	-
B	3	5	5	4.58	0.63	-
A and B	3	5	4.25	4.31	0.61	0.001
PU						
A	2.83	4.83	3.75	3.75	0.51	-
B	2.83	4.66	3.66	3.77	0.61	-
A and B	2.83	4.83	3.66	3.76	0.55	0.279

This data, apparently, shows that the participants perceived DVD as being easy to use and useful considering the mean and medians higher than 3. However, we must verify this result checking the hypotheses. For this, we applied the Shapiro-Wilk test to check the normality of the distribution for both variables (column *p-value* in Table 7.2). The results show that PEOU does not have a normal distribution ($PEOU < 0.05$) and PU has a normal distribution ($PU > 0.05$). To finalize, we applied the Wilcoxon-test in PEOU data and the one-sample T-test in PU data to confirm if the mean is higher than 3 ($PEOU = 0.000$ and $PU = 0.000$). As the results of the Wilcoxon-test and one-sample test were lesser than 0.05, we can accept that DVD was perceived as ease to use and useful, confirming hypotheses H1-a and H2-a.

7.1.3 Threats to validity for the quasi-experiment

Certain issues may threaten the validity of this experiment. About *internal validity*, the main threats are the learning effect, participants' experience, information exchange

among participants, and understandability of the documents. The learning effect was mitigated by ensuring that each group of participants worked with only one experimental object. Participants' experience was not an issue as none of them had previous experience with DVD. To minimize the information exchange among participants, they were monitored by the experimenters to avoid communication biases while performing the tasks. Understandability of the material was alleviated by performing a pilot study and with the translation of the materials to Portuguese (the participants' native language). Concerning *external validity*, the main threats are representativeness of the results and the size and complexity of the tasks. The representativeness of the results may be affected by the business description used and the participants' context selected. About the selection of business description, we mitigated this by considering two different experiment objects with a set of artifacts with similar size and complexity. Regarding size and complexity of the tasks, we used small tasks since an experiment requires participants to complete the assigned tasks in a limited amount of time. The main *construct validity* threats respect the measures applied in the data analysis and the reliability of the questionnaire. We mitigated this by using measures that are commonly used in other software engineering experiments, and the variables are based on TAM [66, 147]. The reliability of the questionnaire was tested with the Cronbach test [184]. Finally, *conclusion validity* threats are the data collection and the validity of the statistical tests applied. About the data collection, we used the same procedure in each experiment to extract the data and ensured that each dependent variable is calculated by applying the same formula. With regard to the validity of the statistical tests proposed, we chose the most common tests employed in empirical software engineering due to their robustness and sensitivity [173].

7.2 Comparing the methods DVD and e3value

This section reports on a family of three controlled experiments carried out to compare the two business value modeling methods e3value [105] and DVD. While e3Value is a well-known method for business modeling [162, 212], DVD was developed in the context of this Ph.D. work to facilitate communication between business and IT stakeholders, and to use business values to drive the development of an information system. Both methods are intended to assist in the alignment between the business and the software development [104, 238] areas, and both represent the basic concepts found in a value model [15]. These two methods were evaluated and compared with respect to their effectiveness, efficiency, perceived ease of use, perceived usefulness and intention to use.

The first controlled experiment was conducted in Spain with MSc students at *Universitat Politècnica de València (UPV)* [241]. The results favored the DVD method, with the exception of the perceived usefulness of the methods, for which no significant difference could be found. This was the major motivation for the exact replication subsequently performed in Portugal with MSc students at *Universidade NOVA de Lisboa (UNL)*. In this case, all the variables evaluated favored the DVD method. The second (exact) replication took

place in Brazil and aimed at validating the results from the two previous experiments with Business Management [Ph.D.](#) students (who are also practitioners in industry) at [Universidade Federal de Pernambuco \(UFPE\)](#). The results of each experiment are analyzed individually and the empirical findings obtained in each experiment are aggregated by means of a meta-analysis.

7.2.1 Experimental methodology

The methodology adopted for the experiments is an extension used by Gonzalez-Huerta *et al.* [101] for the five-steps proposed by Ciolkowski *et al.* [33]. The experiments were designed and executed by following the guidelines proposed by Wohlin *et al.* [265].

Step 1: Experiment preparation Following the [GQM](#) template [22], the goal of our family of experiments is to **analyze DVD** and e3value models and their modeling processes **for the purpose of** comparing them **with respect to** their actual efficacy (effectiveness and efficiency), perceived efficacy (perceived ease of use and perceived usefulness), and intention to use in the future **in order to** obtain high-quality value models **from the point of view of** both business analysts and software engineers, **in the context of** business modelers (novice software engineers and business analysts).

Step 2: Context definition The context of the set of experiments is the quality evaluation of two business models carried out by business modelers. The context is defined by (i) the business model to be evaluated, (ii) the value-driven modeling method, and (iii) the selection of participants. Details on the above are provided in Section [7.2.2](#).

Step 3: Experimental tasks The experimental tasks were structured to allow the comparison of both methods. Depending on the method, each modeling task was composed of the method activities that help to achieve its purpose (e.g., defining a value model). After applying the method, the participants had to fill in a post-experimental questionnaire containing subjective questions regarding their perceptions of the method (see details in Section [7.2.2.3](#)).

Step 4: Individual experiments The family of experiments is summarized in Figure [7.1](#). A baseline experiment ([UPV](#)) [241] was conducted in Spain. It was first internally replicated in Portugal ([UNL](#)) and later externally replicated in Brazil ([UFPE](#)), in order to attain more evidence for the results obtained after carrying out the baseline experiment ([UPV](#)). The external replications allowed us to increase the external validity.

Step 5: Individual data analysis and meta-analysis The results of each individual experiment were collected using a spreadsheet and imported to the *SPSS v20* statistical tool [123], after which they were analyzed individually. We then joined the data and imported

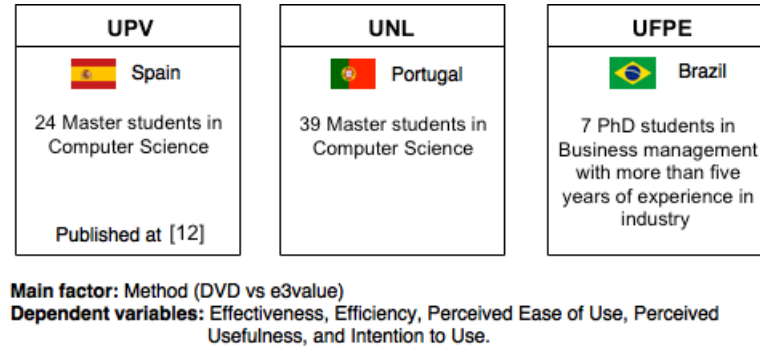


Figure 7.1: Overview of our family of experiments.

it to the *R Studio* tool [248] in order to perform the meta-analysis. The analysis procedure is detailed in Section 7.2.2.6.

7.2.2 Baseline Experiment

An initial design for this experiment was presented and discussed in a workshop held at the ACM/IEEE MODELS conference [238]. The feedback received during the discussion (for example, about the process used to measure the effectiveness of the participants) was taken into account and incorporated into the baseline experiment. The following subsections define the research questions and hypotheses, the sample and participants, the experimental objects and tasks, the metrics and design, and, finally, the analysis procedure of the experiment.

7.2.2.1 Research questions and hypotheses

There are two research questions addressed by the family of experiments:

RQ1 Which of the methods has the higher actual efficacy, e3value or DVD?

RQ2 Is the perceived efficacy and intention to use of the participants favoring e3value or DVD?

The independent variable of interest is the use of each value-driven modeling approach with nominal values: DVD and e3value. Two treatments were, therefore, employed in the experiment: the creation of a value model for two software systems using the DVD method and the creation of a value model for the same systems using the e3value method. The experimental data collected made it possible to compare the effects of both treatments. Figure 7.2 shows the taxonomy of the types of dependent variables used in this experiment.

There are two types of dependent variables in which the treatments are compared: performance-based and perception-based. Performance-based variables assess how well the participants perform the experimental task. They are used to evaluate the actual efficacy of the methods. Perception-based variables assess the participants' perceptions of their performance and their subsequent intention to use the methods DVD or e3value.

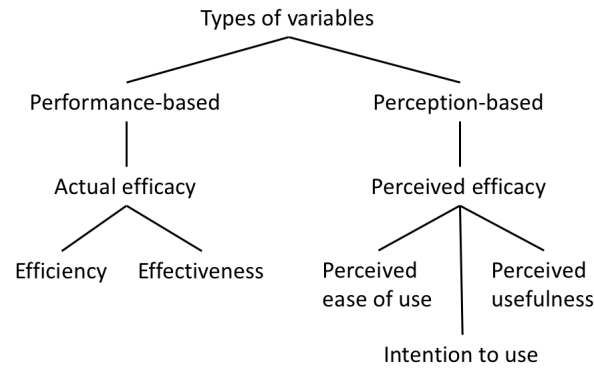


Figure 7.2: Taxonomy of dependent variables.

These variables are used to evaluate the perceived efficacy of the methods, and their likely adoption in practice. There are two performance-based variables:

- *Efficiency*, measuring the modeling time (i.e., the time required to apply the method).
- *Effectiveness*, measuring the correctness and completeness of the value model created by the participants.

There are also three perception-based variables: *Perceived Ease of Use*, *Perceived Usefulness*, and *Intention to Use*. These variables were identified using the TAM [66], a widely applied theoretical model used to analyze user acceptance and usage behavior as regards emerging information technologies through the use of empirical validations and replications [147]. The perceived efficacy [66] of the method can, therefore, be decomposed into the following subjective dependent variables:

- *Perceived Ease of Use (PEOU)*, indicating the degree to which a person believes that learning and using a particular value-driven method would occur with reduced effort.
- *Perceived Usefulness (PU)*, indicating the degree to which a person believes that using a particular method will increase her/his job performance within an organization.
- *ITU*, indicating the extent to which a person intends to use a particular method. It represents a perceptual judgment of the method's efficacy, that is, whether it is cost-effective and is commonly used to predict the likelihood of acceptance of a method in practice.

We formulated several null hypotheses, which were defined in a one-tailed manner since we wished to analyze the effect of the use of value-driven methods on the described variables. Each null hypothesis and its alternative are presented as follows:

- H1-0: There is no significant difference between the effectiveness of the DVD and e3value methods / H1-a: The DVD method is significantly more effective than the e3value method.

- H2-0: There is no significant difference between the efficiency of the DVD and e3value methods / H2-a: The DVD method is significantly more efficient than the e3value method.
- H3-0: There is no significant difference between the perceived ease of use of evaluators applying the DVD and e3value methods / H3-a: The DVD method is perceived as easier to use than the e3value method.
- H4-0: There is no significant difference between the perceived usefulness of the DVD and e3value methods / H4-a: The DVD method is perceived as more useful than the e3value method.
- H5-0: There is no significant difference between the intention to use the DVD and e3value methods / H5-a: The DVD method is perceived as more likely to be used than the e3value method.

Note that although we have no reason to believe that one method is better than the other, the formulation of the hypothesis starts with the DVD method by chance, and we could have chosen the e3value method to start those formulations.

7.2.2.2 Sample and participants

The sample in the baseline study is a group of 24 MSc students at the UPV, in Spain. The experiment was a class exercise on an “Empirical Software Engineering” course, which included an introduction to the e3value and DVD methods. The participants had no previous experience of value-driven modeling methods before attending this course. However, they had previous experience in modeling software with the UML and had an average of three years experience in software development.

7.2.2.3 Experimental objects and tasks

Two experimental objects were selected from the following two software requirements systems in literature [69, 122]:

- Wireless access provisioning (Object1): a hotel offers wireless connectivity to businessmen as an additional service. Such service must be provided as a joint service offering of the hotel. In other words, the businessmen pays the hotel for both the room and the wireless access service and the hotel determines the fee for the wireless service.
- Waste management (Object2): waste is traded between an exporter and an importer. The exporter usually pays the importer for the waste handling, but in some cases, the waste can be traded like a regular good, such as recycled waste.

The size of these two experimental objects is comparable. The experimental task was to create a value model following the specific steps of each method (e3value or DVD). Figure 7.3-a summarizes the process employed to create an e3value model [104].

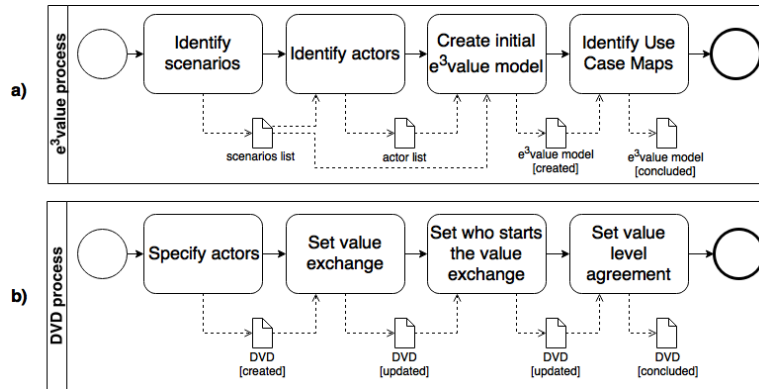


Figure 7.3: Processes for the creation of (a) an e3value model and (b) a DVD model.

Participants had to identify a list of scenarios (or short textual descriptions of the product, service, or experience expected by a customer), after which they had to identify the actors (who offers and who receives the product, service or experience expected) from the list of scenarios. They then had to create the initial e3value model using the products and services mentioned in the list of scenarios and the actors in the list of actors, and add the macro activities in order to operationalize the value exchange. Finally, they had to insert the UCM elements representing the paths of all the scenarios.

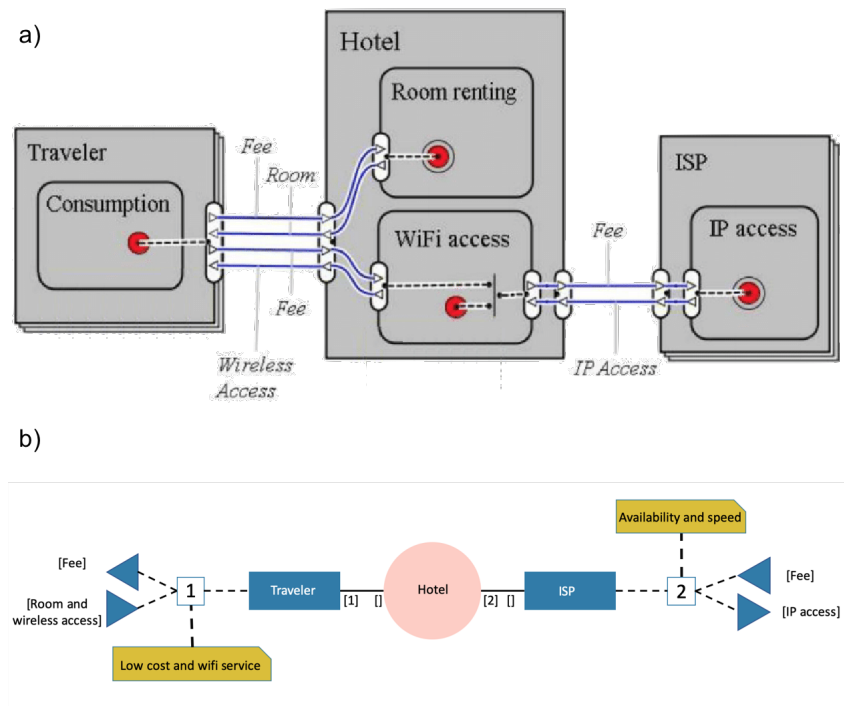


Figure 7.4: Initial oracles of (a) e3value and (b) DVD for Object1.

Similarly, Figure 7.3-b shows the process of creating a DVD model. It starts by describing the actors: main actor and environment actors. Using the idea of “main actor” as the focus of the analysis signifies that, for a complex system, the designer may be the need to build as many DVD models as the number of the actors that are fundamental to the business so that the whole business is studied. Participants were asked to build a single DVD model, after which they had to add the value exchanges to the model, define the value elements related to each value port and continue by determining which actor originates the value exchange, checking whether the value elements are specified in the correct value port. The final step is to define the criteria required for value exchanges to be performed and it is crucial to understand the business constraints related to each value exchange.

The expected final model for each of the experimental objects are easily modeled in both DVD and e3value methods. For example, Figure 7.4 shows the expected final models obtained for the methods e3value (a) and DVD (b) for the Object1 case. These expected final models are used as a baseline (Oracle) to measure the models created by the participants (details are given in subsection 7.2.2.4).

Once the value model was created, the participants answered the post-experimental questionnaire [242]. This questionnaire, defined as a Google Form, contains a set of closed-questions, allowing the participants to express their opinion on the ease of use, usefulness, and intention to use the method in the future. It also includes three open questions to collect the participants’ feedback regarding the changes they would make to improve the method and their reasons to use one or the other method in the future (if any). The data collected was kept anonymous.

The answers to the questionnaire were the base to evaluate the perception-based variables (PEOU, PU, and ITU). The performance-based variables (effectiveness and efficiency) were evaluated by comparing the value model created by the participants with the value model designed by experts and by analyzing the time required to perform each experimental step.

7.2.2.4 Metrics

We used an approach based on the information retrieval theory [94] to obtain a quantitative assessment of the Effectiveness of value models modeled with both the methods e3value and DVD. This same approach has been applied in other software engineering experiments [4, 221] to compare models created by participants with an Oracle (the correct model created by an expert) regarding each type of graphic elements, using equations (7.1) and (7.2) for *precision* and *recall*, in which the $precision_{element}$ measures the correctness of a graphical element belonging to a given value model and the $recall_{element}$ measures the completeness of a value model as regards to its graphical element.

$$precision_{element} = \frac{|P_{element} \cap O_{element}|}{|P_{element}|} \quad (7.1)$$

$$recall_{element} = \frac{|P_{element} \cap O_{element}|}{|O_{element}|} \quad (7.2)$$

Accordingly, $P_{element}$ indicates all the particular graphical elements modeled by a participant and $O_{element}$ represents the known correct set of expected types of graphical elements that can easily be derived using an Oracle.

Precision and recall quantitatively summarize two different concepts. We therefore used their harmonic mean [94] to obtain a balance between the correctness and completeness of each graphical element in a value model (equation 7.3):

$$F - Measure = \frac{2 * precision_{element} * recall_{element}}{precision_{element} + recall_{element}} \quad (7.3)$$

The F-Measure quantitatively summarizes the accuracy of a value model as regards its graphical elements and is compared with an Oracle.

The effectiveness dependent variable is computed as the arithmetic mean of the entire F-Measure. All the measures above assume values of between 0 and 1. Whatever the measure is, 0 is the worst value and 1 is the best. With regard to effectiveness, values close to 1 signify that the participants defined value models that were very similar to the Oracle. Conversely, values close to 0 indicate that the models were very different from the Oracle. The effectiveness variable has been defined in order to give the same relevance to the correctness and completeness of value models for all the graphical elements of the value model.

The first Oracle was developed by an expert in value modeling before the experiment (one for each experiment object as can be seen in the Figure 7.4). In the case of e3value, the first Oracle was extracted from the literature [122], [69]. As value models could have different levels of granularity, the expert developed new Oracles with different levels of abstraction. For example, in the Oracle represented in Figure 7.4-a, the participants could create only one activity to represent all hotel services (e.g., “hotel services” activity within the Hotel actor rather than creating the “Room renting” and “WIFI access” activities). At the end, we checked the effectiveness of all models created by the participants against the Oracles, and the higher effectiveness result was selected.

The three subjective variables (e.g., PEOU, PU, and ITU) were measured using a 5-point Likert scale questionnaire with a set of 12 closed-questions: 5 questions for perceived ease of use (PEOU), 5 for perceived usefulness (PU), and 2 for intention to use (ITU) [242]. These were formulated using the opposing statement format, signifying that each question contains two contradictory statements representing the maximum and minimum possible values (5 and 1), where 3 is considered to be a neutral perception². The aggregated value of each variable was calculated as the arithmetical mean of the answers

²It is common for participants not committed to the research to try to complete the questionnaire by answering all questions with 5 (in favor of the method) or 1 (against the method). In order to avoid this bias, we use two contradictory statements, hence canceling out the result of these questions for this participant.

to the questions associated with each perception-based variable. We used Cronbach's alpha test to evaluate the reliability of the survey and of each variable.

7.2.2.5 Design and execution

The experiment was planned as a balanced within-participant design with a confounding effect, i.e., the same participants would apply both methods with both experimental objects in a different order. We formed two groups (each of which used one method to one experimental object) to which the participants were randomly assigned. Table 7.3 summarizes the design of the experiment. The within-participant experimental design is intended to minimize the impact of learning effects on the results since none of the participants repeats any treatment or experimental object during the execution. The comprehension of the software systems requirements may also have affect the application of both methods. We alleviated the influence of this factor by selecting two representative software systems with requirements of a complexity suitable for application in the time slot available for the execution of the experiments (2 sessions of one hour each).

Table 7.3: Experiment design

Groups	Session 1	Session 2
A	Object1, e3value	Object2, DVD
B	Object1, DVD	Object2, e3value

We conducted a pilot experiment with 2 professors and 1 Computer Science Master's Degree student at the UPV. They played no further part in the controlled experiments. The goals of this pilot experiment were to evaluate all the experimental material, the instructions regarding the experimental procedure and the task completion time. The results indicated that the experiment objects were well suited and that one hour were sufficient to accomplish the task. No software tool was used during the execution of the experiments, to avoid possible usability bias.

A training session explaining the concepts and processes was provided to the participants, who had to create a value model by following the experimental procedure. During the experiment session, the participants were given a pencil, an eraser, sheets of paper and the printed copy of the experimental material slides introducing business modeling and value-driven development, slides describing the value-driven development method and an application example, the slides describing the e3value and DVD methods with an example, the specification documents of the software systems to be used in the tasks, and the post-experimental questionnaire. The material was in the participants' native language (e.g., Spanish). No interaction among participants was allowed and no time limit by which the tasks had to be completed was imposed. Moreover, we provided no details on how to deal with the modeling tasks, but any issues concerning the specification documents were clarified. Finally, the participants were asked to register their start

and end times for each step performed. The answers to this questionnaire were the basis employed to evaluate the perception-based variables (perceived ease of use, perceived usefulness, and intention to use).

The performance-based variables (effectiveness and efficiency) were evaluated by comparing the value model they created with the value model designed by the expert and by analyzing the time required to perform each experimental step.

7.2.2.6 Analysis procedure

We chose to analyze the data collected with statistical tests owing to their robustness and sensitivity and because they have been used in similar experiments ([46], [4]). As is usual, we accepted a probability of 5% of committing a Type-I-Error [265] in all the tests, i.e., rejecting the null hypothesis when it is true. We tested the normality of the data distribution by applying the Shapiro-Wilk test. The results of the normality test allowed us to select the correct significance test with which to examine our hypotheses. When data was assumed to be normally distributed ($p\text{-value} > 0.05$), we applied the parametric one-tailed t-test for independent samples [138]. However, when data did not assume the normal distribution ($p\text{-value} < 0.05$), we applied the non-parametric Mann–Whitney test [61].

7.2.3 Discussion of results

The results obtained in the baseline experiment show that the values for all variables are higher for the DVD method (see Table 7.4). Before applying the analysis procedure (Section 7.2.2.6) in order to confirm the results, we used the Cronbach’s alpha to examine the reliability of the questionnaire. The test result for Cronbach’s alpha for the whole questionnaire was 0.928 and that for each variable was 0.889 (PEOU), 0.802 (PU), and 0.850 (ITU), signifying that the questionnaire is very reliable (i.e., Cronbach’s alpha is

Table 7.4: Descriptive statistics for effectiveness, efficiency, PEOU, PU, and ITU per experiment and method.

Experiment	Number of observations	Variable	e3Value					DVD				
			Min.	Max.	Med.	Mean	Std. Dev.	Min.	Max.	Med.	Mean	Std. Dev.
UPV	48	Effectiveness	0.26	0.75	0.55	0.56	0.11	0.50	1	0.87	0.83	0.14
		Efficiency	15	56	30.50	33.08	10.85	6	37	16.50	20.04	9.89
		PEOU	1.6	5	3.40	3.41	0.87	1.2	5	4.70	4.25	0.99
		PU	1.8	5	3.40	3.29	0.66	1.6	5	3.80	3.66	0.95
		ITU	1	5	3.25	3.10	1.09	1	5	4.00	3.75	0.96
UNL	78	Effectiveness	0.22	0.82	0.55	0.54	0.14	0.14	1	0.81	0.75	0.22
		Efficiency	7	64	25	29.33	15.89	4	49	20	21.72	13.24
		PEOU	2	4.25	3.25	3.10	0.61	3	5	4.25	4.31	0.61
		PU	2.16	4.33	3.33	3.36	0.58	2.83	4.83	3.66	3.76	0.55
		ITU	1.5	4.5	3	3.19	0.74	2	5	3.5	3.73	0.87
UFPE	14	Effectiveness	0.11	0.65	0.43	0.44	0.18	0.33	1	0.83	0.73	0.28
		Efficiency	24	63	30	38.71	16.55	6	42	23	21.57	12.20
		PEOU	1	4.25	3	2.92	1.36	2.75	5	4.75	4.32	0.82
		PU	1	4.2	3.16	2.85	0.99	3.2	5	3.50	4.04	0.79
		ITU	1	5	3.50	3	1.29	3	5	3.50	3.64	0.97

higher than 0.7 [184]), indicating that the questionnaire is not biased as regards the perceived-based variables.

Figure 7.5 shows the analysis procedure used to confirm the results. We first applied the Shapiro-Wilk test to verify the normality of the distribution of all variables (effectiveness=0.108, efficiency=0.052, **PEOU**=0.000, **PU**=0.465, and **ITU**=0.005). The results show that effectiveness, efficiency, and **PU** have a normal distribution ($p\text{-value} > 0.05$). We therefore applied a $t\text{-test}$ (parametric test) to verify hypotheses H1-0 (effectiveness), H2-0 (efficiency), and H4-0 (**PU**) and a Mann-Whitney test (non-parametric test) to check hypotheses H3-0 (**PEOU**) and H5-0 (**ITU**).

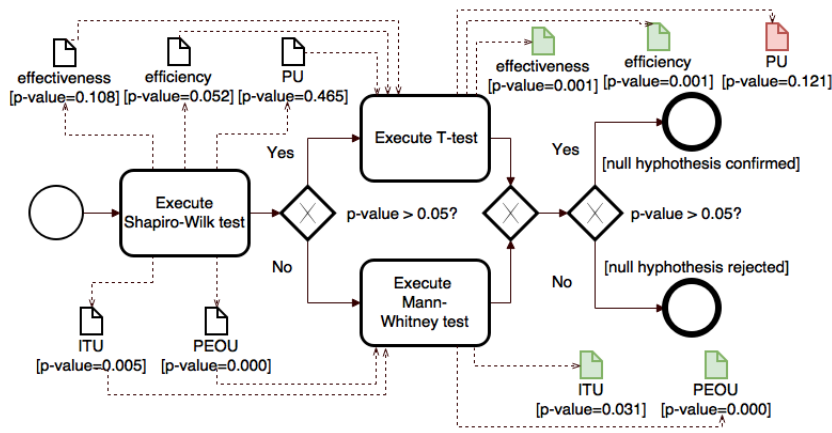


Figure 7.5: Analysis procedure employed for the experiment in Spain (UPV).

The $p\text{-value}$ results obtained from the $t\text{-test}$ were effectiveness=0.001, efficiency=0.001, and **PU**=0.121. As the $p\text{-value}$ for **PU** is higher than 0.05, we can confirm hypothesis H4-0, meaning there is no significant difference between the methods. Null hypotheses H1-0 and H2-0 must, however, be rejected because the $p\text{-values}$ for effectiveness and efficiency are lower than 0.05. With regard to **PEOU** and **ITU**, the results for the Mann-Whitney test were 0.000 and 0.031, respectively. As both results are lower than 0.05, we cannot confirm hypotheses H3-0 and H5-0, showing that the participants perceived the **DVD** method to be easier to use than the e3value method (thus confirming H3-a) and their intention to use **DVD** in the future is higher than that of using e3value (thus confirming H5-a). In summary (see Figure 7.5), only the result obtained for **PU** (H4-0) confirms the null hypothesis (artifact in red).

With regard to the RQ1 (*Which of the methods has the higher actual efficacy, e3value or DVD?*), the data analysis results indicate a significant difference between the methods concerning efficiency (time required to create the model) and effectiveness (correctness and completeness of the model). One plausible justification for this result is that **DVD** facilitates the representation of the business economic point of view, thanks to its cognitive-based, semi-structured nature. With regard to RQ2 (*Is the perceived efficacy and intention to use of the participants favoring e3value or DVD?*) the data analysis results show that the perceived efficacy is higher for the **DVD** method. However, the results show no significant

difference between the methods for perceived usefulness (PU). This is not surprising as both methods share the same goal and represent the same central economic concepts. In the case of perceived ease of use (PEOU), the results indicate that the DVD method is significantly easier to use than the e3value method. We also associate this result with the DVD method being structured as a cognitive mind map.

7.2.4 Two Experimental Replications

Two experimental replications of the previous controlled experiment were performed, one in Portugal and another in Brazil.

These were needed for two reasons. First, the null hypothesis H4-0 (*there is no significant difference between the perceived usefulness of the DVD and e3value methods*) could not be rejected in the (baseline) experiment performed in Spain, meaning that the participants perceived both methods to be equally useful for defining value models. As the descriptive statistics analysis shows that the PU result for the DVD method is higher than that of the e3value method, we believed that H4-0 would be rejected if we increased the number of participants. We consequently performed a replication with more participants at the UNL in Portugal. Secondly, we felt the need to execute an experiment with experienced participants with a business background in order to verify whether the results would hold, thus increasing the validity of the results. This replication was performed at the UFPE in Brazil. It is essential to highlight that, with the exception of the experimental material which was translated into the participants' native language (e.g., Portuguese-PT and Portuguese-BR), we did not change any of the experimental conditions of the experiment conducted in Spain. These experiments are, therefore, exact replications of the baseline experiment.

7.2.4.1 Sample and participants

The sample in the replication was composed of 46 participants: 39 MSc students in Computer Science in Portugal and 7 Business Management Ph.D. students in Brazil. The 39 MSc students were attending the "Software Engineering" and "Requirements Engineering and Software Architecture" courses at UNL. These participants had no previous experience with value-driven modeling methods, but they were experienced in software modeling. In particular, they were familiar with UML and had an average of three years of experience in software development. The experiment took place during April 2017. The 7 Business Management Ph.D. students in Brazil were attending the "Business Process Modeling" course at the UFPE. Before attending this course, these participants had theoretical knowledge of value modeling (e.g., REA [174] and BMO [191]), but no previous experience in the methods used in the experiment. It is worth noting that the Brazilian experiment is important because all the participants are also professionals from industry with more than five years of experience. Despite the small number of participants, the experiment had a balanced within-participant design, what means that the number of

observations generated is double the number of participants. The experiment took place during June 2017.

7.2.4.2 Results

This section discusses the results from the replications performed in Portugal (UNL) and Brazil (UFPE).

Internal Replication (UNL): Similarly to the results obtained in the UPV experiment, the descriptive statistics results for all the variables of the UNL experiment also favor the DVD method (see Table 7.4). Again, Cronbach's alpha was used to examine the reliability of the questionnaire, and the result obtained for the questionnaire was: PEOU questions = 0.803, PU questions = 0.705, ITU questions = 0.732, while that for the whole questionnaire = 0.858. This means that the questionnaire can be considered reliable (Cronbach's alpha is higher than 0.7 [184]).

Figure 7.6 shows the analysis procedure used to confirm the results of this experiment. After using this process, the Shapiro-Wilk test resulted in the following values for each variable: effectiveness=0.077, efficiency=0.001, PEOU=0.005, PU=0.407, and ITU=0.005). Given that $p\text{-value} > 0.05$ for effectiveness and PU, we concluded that the effectiveness and PU data had a normal distribution, so we could apply a parametric statistical test to analyze them. However, it was necessary to apply a non-parametric statistical test to analyze the remaining variables.

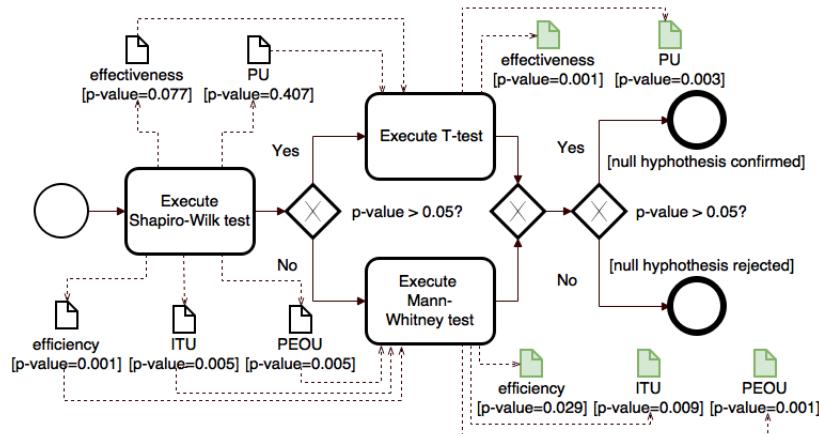


Figure 7.6: Analysis procedure employed for the experiment in Portugal (UNL).

We applied a t -test to compare the results obtained for effectiveness (0.001) and PU (0.003). These results allowed us to reject hypotheses H1-0 and H4-0 ($p\text{-value} < 0.05$), meaning that the participants obtained higher quality value models when applying DVD and that they perceived it to be more useful for creating value models than the e3value method.

With regard to the efficiency, PEOU and ITU variables, the non-parametric test used to compare the results was the Mann-Whitney test. The results were efficiency=0.029,

$PEOU=0.001$, and $ITU=0.009$. This allowed us to reject hypotheses H2-0, H3-0, and H5-0 because $p\text{-value}<0.05$ means that participants created the DVD models significantly faster than the e3value models. The DVD method was also perceived to be considerably easier to use than the e3value method and the participant's intention to use DVD in the future was substantially higher than that of using e3value. Overall, these results confirm that the participants were more efficient and effective when using the DVD method. Unlike the baseline experiment, the results for all the variables in Portugal favored the DVD method, and both research questions (e.g., RQ1 and RQ2) obtained positive responses.

External Replication (UFPE): The descriptive statistics results obtained for the UFPE experiment show that the DVD method is better ranked in all variables (see Table 7.4. The results of Cronbach's alpha show that the questionnaire is reliable (all questionnaire=0.952, PEOU questions=0.939, PU questions=0.920, and ITU questions=0.784), as Cronbach's alpha is higher than 0.7 [184], thus allowing us to apply the analysis procedure. Figure 7.7 shows the analysis procedure used to confirm the results of this experiment.

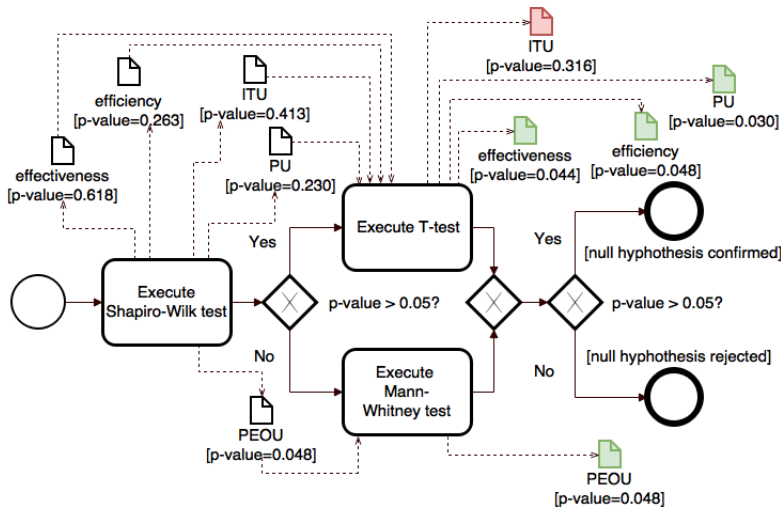


Figure 7.7: Analysis procedure employed for the experiment in Brazil (UFPE).

The Shapiro-Wilk test was used to verify the normality of the distribution for each variable. The results (effectiveness=0.618, efficiency=0.263, $PEOU=0.048$, $PU=0.230$, and $ITU=0.413$) show that all the variables have a normal distribution, with the exception of $PEOU$ which has a $p\text{-value}<0.05$. The non-parametric Mann-Whitney test was consequently applied in order to verify hypothesis H3-0 ($PEOU$), while the parametric $t\text{-test}$ was applied to verify hypotheses H1-0 (effectiveness), H2-0 (efficiency), H4-0 (PU), and H5-0 (ITU). The result of the tests was: effectiveness=0.044, efficiency=0.048, $PEOU=0.048$, $PU=0.030$, and $ITU=0.316$. All the variables, with the exception of ITU , have a $p\text{-value}<0.05$, signifying that the null hypotheses can be rejected and that the alternative hypotheses H1-a, H2-a, H3-a, and H4-a confirmed. In others words, the results show that the participants were more effective and efficient when using the DVD

method and they also perceived DVD to be easier to use and more useful than the e3value method. With regard to ITU, as the result obtained from the test was higher than 0.05, we cannot confirm hypothesis H5-a, meaning that there is no significant difference between the participants' intention to use these methods (although the mean value obtained for the DVD method is higher than that obtained for e3value). In summary, the results of this experiment show that DVD was considered a better alternative with respect to the variables analyzed, with the exception of ITU.

7.2.5 Meta-Analysis

Among the existing statistical methods to aggregate results from interrelated experiments [116, 215], meta-analysis allows more general conclusions to be obtained and was, therefore, chosen for this study. Meta-analysis is a set of statistical techniques that can be used to combine and contrast the results (e.g., patterns and sources of disagreement) of multiple scientific studies [216]. Figure 7.8 shows the forest plot (or blobbogram) provided by the R Studio tool [248] used. The square expresses the magnitude of the effect of the method while the dimensions of the square are proportional to both the weight of the experiment in the meta-analysis and the number of participants. The result for studies with a large sample size is more accurate, meaning that they make a greater contribution to the overall effect [4]. The effect size obtained in our meta-analysis varies between small and medium in all cases. This may indicate that it will be necessary to perform further replications with a larger sample of participants. Despite this, and given that no other similar studies exist in the literature, the present results are still useful and of interest to the community.

The confidence intervals of each experiment are represented by horizontal lines. We considered a confidence interval of 95 percent for each experiment. When these horizontal lines cross over the central vertical line of the graph, this means that there is no significant difference between the means of the methods (e.g., PU in the experiment conducted in Spain and ITU in the experiment conducted in Brazil). The diamonds represent the overall conclusion. The summary measure is the central line of the diamond, while the associated confidence interval is the lateral tips of the diamond. When the diamond crosses over the central vertical line of the graph, this means that there is no significant difference between the aggregated result. As this did not occur in our meta-analysis, the aggregated result was, therefore, always favorable for one of the methods.

Despite the fact that the null hypotheses H4 (related to PU) and H5 (related to ITU) could not be confirmed in the UPV and UFPE experiments, the overall results of the meta-analysis have a significant positive effect. The diamonds are always positioned on the DVD method side (for example, on the right-hand side of the effectiveness graph) and we can, therefore, reject all null hypotheses. In summary, the meta-analysis strengthens the results obtained in the individual experiments.

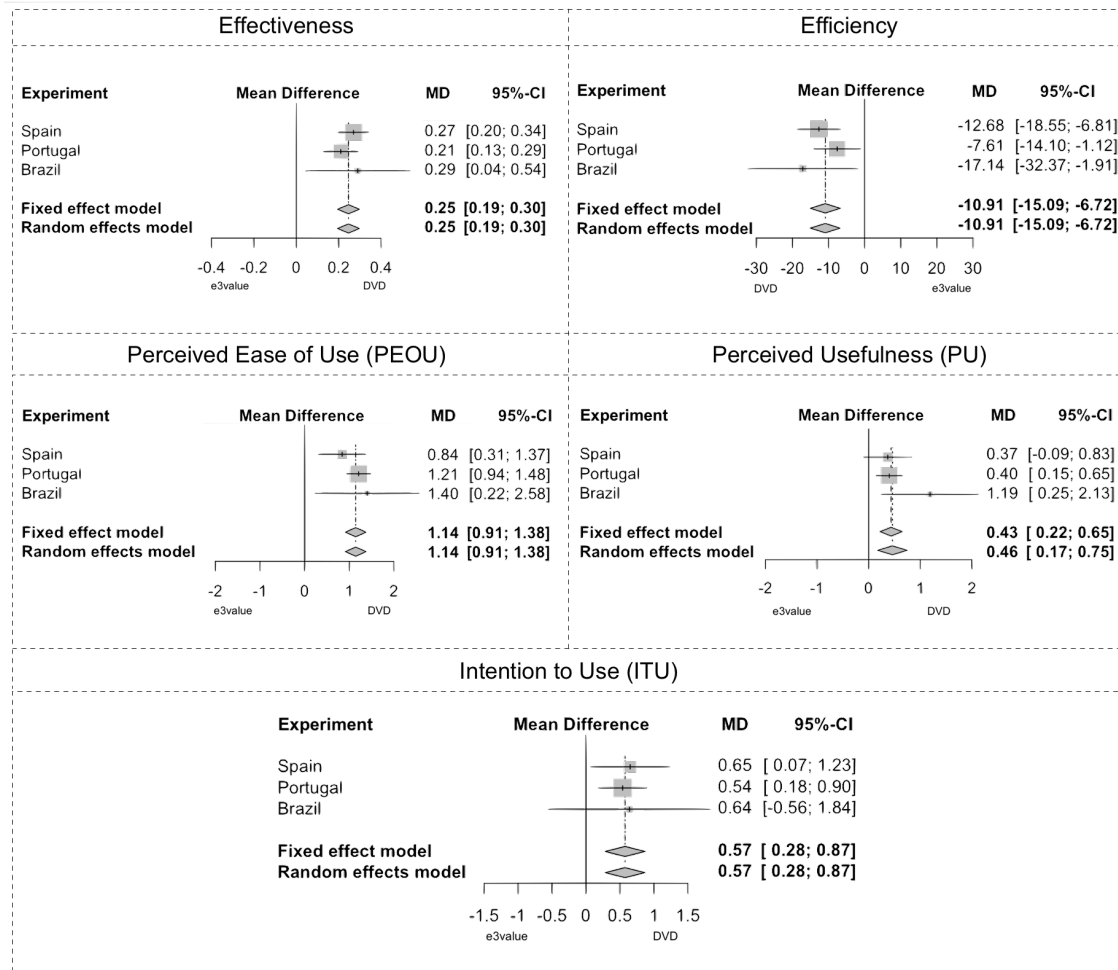


Figure 7.8: Meta-analysis blizzard diagram for effectiveness, efficiency, PEOU, PU and ITU.

7.2.6 Discussion of the results

Figure 7.9 summarizes the descriptive statistic results for the three experiments. The small number of outliers were discarded from the data analysis. These outliers occurred because some participants did not participate in the training session, or arrived late. They just attended the review that was held before each experimental section.

Table 7.5 summarizes the results for the various hypothesis (where an accepted null hypothesis means no significant difference between e3value and DVD, and an accepted alternative hypothesis means that the result favors DVD). Besides, we calculated the value of *Cohen's d* [59] and the effect-size correlation (*effect size r*) [184] using the means and standard deviations of two groups (treatment and control)³. The sign of our *Cohen's d* effect (*Cohen's d* column in Table 7.5) indicates the direction of the effect. In the case, the negative sign means that the direction of the effect is in favor of the DVD method. Note that *Cohen's d* result for efficiency is positive, meaning that the participants took longer to model using e3value. In other words, the least efficient method is the one

³Details on how to calculate *Cohen's d* and *effect size r* can be found in [29, 30, 59].

7.2. COMPARING THE METHODS DVD AND E3VALUE

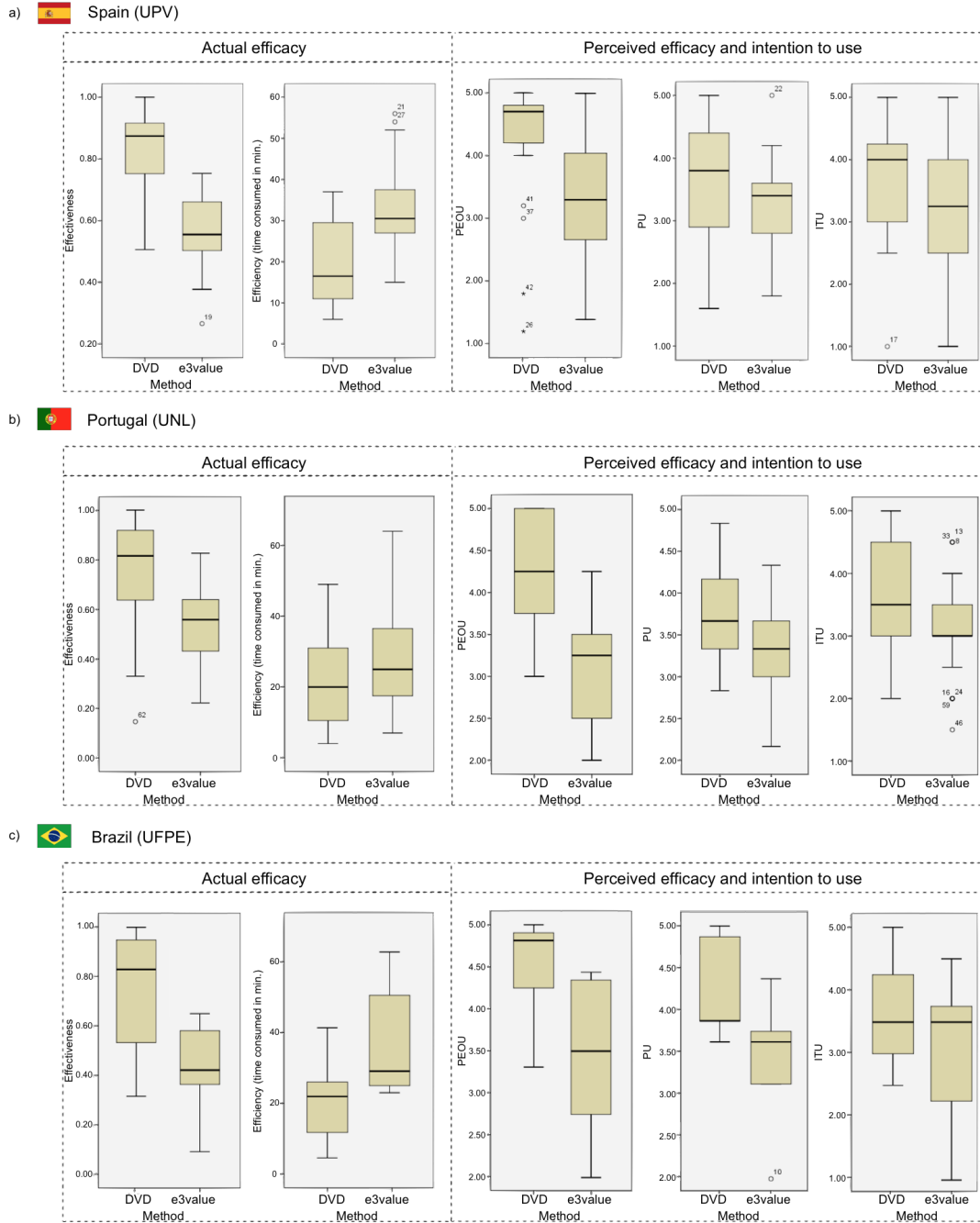


Figure 7.9: Actual efficacy (effectiveness and efficiency), perceived efficacy (PEOU and PU), and ITU grouped by methods of experiments performed in (a) Spain, (b) Portugal, and (c) Brazil.

that has the positive result. Regarding the *effect size* r , Cohen provided rules for their interpretation, suggesting that an *effect size* r between $|.10|$ and $|.29|$ represents a “small” effect size, between $|.30|$ and $|.49|$ represents a “moderate” effect size, and larger than $|.5|$ represents a “large” effect size [59]. The last column in Table 7.5 shows that the effect size of our experiments is, mainly large and moderate. Even though we have some results

with small effect size, we believe that the results of our family of experiments are still relevant to the community because there are no other works that empirically compare value-driven development methods.

Table 7.5: Summary of the results of all experiments, where checkmark means hypothesis accepted and X means hypothesis rejected.

Experiment	Variable	Null hypothesis	Alternative hypothesis	Cohen's d	Effect size r	Effect size interpretation
UPV	Effectiveness	✗	✓	-2.14	-0.73	Large
	Efficiency	✗	✓	1.25	0.53	Large
	PEOU	✗	✓	-0.90	-0.41	Moderate
	PU	✓	✗	-0.45	-0.22	Small
	ITU	✗	✓	-0.63	-0.30	Moderate
UNL	Effectiveness	✗	✓	-1.13	-0.49	Moderate
	Efficiency	✗	✓	0.52	0.25	Small
	PEOU	✗	✓	-1.98	-0.70	Moderate
	PU	✗	✓	-0.70	-0.33	Moderate
	ITU	✗	✓	-0.66	-0.31	Moderate
UFPE	Effectiveness	✗	✓	-1.23	-0.52	Large
	Efficiency	✗	✓	1.17	0.50	Large
	PEOU	✗	✓	-1.24	-0.52	Large
	PU	✗	✓	-1.32	-0.55	Large
	ITU	✓	✗	-0.56	-0.26	Small

Which of the methods has the higher actual efficacy, DVD or e3value? (RQ1)

The descriptive statistics for effectiveness and efficiency indicate that the **DVD** method performs better than the **e3value** method in the experiments performed in Spain, Portugal, and Brazil. The meta-analysis for the aggregated experiments results confirm a significant difference between the methods regarding *efficiency* (time required to create the model) and *effectiveness* (correctness and completeness of the model). One plausible justification for this conclusion is that the **DVD** method facilitates the representation of the business from an economic point of view, thanks to its cognitive-based, semi-structured nature. Moreover, the **DVD** method has fewer concepts which might also have a positive effect on the modeling time and the participants' perceived ease of use. The responses for the open questions from the questionnaire indicated that the **e3value** method has a weak separation of concerns [73]; it represents static (e.g., objects) and dynamic (e.g., scenarios) business concepts in the same model, thus making the value model complex and arduous to build. Furthermore, the participants indicated that “*DVD is very simple, intuitive, and easy to use*”, or “*I would use it [DVD] because it is not difficult to understand and it would be simple to explain to my clients, saving time in modeling this business point of view*”, or “*It [DVD] is not hard to understand and it uses a simple structure*”, or still “*[DVD] makes the business model construction an effective and fast step*”.

In summary, the **DVD** method appears to represent the essential business value concepts in a structured manner, thus making it a concise technique. The **DVD**'s structure is based on mind map diagrams and inherits the well-known benefits of this structure (e.g., organization, use of keywords, association, grouping ideas, visual memory, and simplicity [51]). The consequence of being concise and having a simple structure seems

to help DVD attain more positive results than e3value. (Note that efficiency in Figure 7.8 may seem misleading. This is because efficiency is measured in terms of modeling time, meaning that the larger the result, the less efficient the method is, which is why the result for efficiency may appear to be the opposite).

Is the perceived efficacy and intention to use of the participants favoring e3value or DVD?

The result of the descriptive statistics analysis of the replications are in line with the experiment baseline (at UPV). Upon considering the analysis of the hypotheses, the results of the replication contradicts the baseline experiment in relation to PU and ITU. With regard to the PU, we did not confirm a significant difference between DVD and e3value in the UPV experiment (H4-0 was confirmed). However, we believed that H4-0 would be rejected if we increased the number of participants because the analysis of the descriptive statistics in the baseline experiment favors the DVD method. The results of the UNL replication confirmed what we believed, in other words, H4-0 was rejected (the DVD method is perceived as significantly more useful than the e3value method). In the UFPE replication, we changed the participants' background from Computer Science to Business Management. The result for PU in this replication is also favors DVD, and the reason might be that no prior IT knowledge is required to create a DVD model.

The meta-analysis confirmed that, despite the result obtained in Spain (UPV), DVD is perceived to be more useful than e3value. One plausible justification for this result is that the DVD method also facilitates the extraction of business knowledge in order to design information systems [239, 240].

One interesting finding that we have identified after carrying out the UFPE replication is that the different backgrounds of the participants (e.g., Computer Science and Business Management) did not significantly alter the results of the experiments. Only the ITU result contradicts those of the other experiments (e.g., UPV and UNL), being significantly higher for DVD in Spain (UPV) and Portugal (UNL). However, this was not confirmed in Brazil (UFPE), despite the fact that the mean obtained for DVD (3.64) was slightly higher than the one obtained for e3value (3). As the analysis of the descriptive statistics in the UFPE replication shows that the ITU result for DVD is higher than for e3value, we believe that H5-0 (*there is no significant difference between the intention to use the DVD and e3value methods*) would be rejected if we were to increase the number of participants with a business background.

In addition, even when considering the result obtained in Brazil, the aggregated results of the experiments confirm that the participants have the intention to use DVD in the future (when appropriate). Given that the Brazilian participants are practitioners, they suggested that the DVD method needs a supporting tool and integration with business processes (e.g., BPMN [263]) to represent the value stream throughout the business activities. With regard to the integration issue, we would like to emphasize that a DVD

model provides a point of view of the business. It needs to be complemented with other models (e.g., process models or goal models) for a more complete representation of the whole business. It is worth highlighting that the DVD method follows a model-driven approach and provides model transformations to the BPMN model [263] (and also to KAOS [65], iStar [268] and SOA services) [239, 240], but this was not part of the experiment. Moreover, even though the e3value method represents value streams (using UCM elements), the result of the descriptive statistics for ITU favored the DVD method.

The questionnaire's open questions also show that the likelihood of intention to use the methods in the future is probably related to the easiness of using the method, as per answers like *"it [DVD] is easier to use and fast to create. Because of this, I would use it in the future"*, or *"I would not use it [the e3value] because it requires a lot of effort to modeling, and provides a complex and confusing diagram. The cost benefit does not pay. I would use it, if it was simpler and more objective"*.

For perceived ease of use (PEOU), results show that DVD is significantly easier than e3value in all the experiments. We also associate this result with the mind map roots of the DVD model. This conclusion is reinforced by the positive answers obtained from the participants' questionnaire, such as, *"(DVD) is easy to understand, simple, and clearly shows those who make the most important exchanges"* and *"I would use this method thanks to its simplicity regarding its use and understanding by non-expert users"*.

Nevertheless, the responses to these open questions also indicated that the participants had some difficulties in understanding the meaning of some modeling elements (e.g., *"[I would like to advise against] using such complicated symbols"*). We plan to perform a new empirical study with the aim of defining a more representative iconography for these methods based on Moody's physics of notation theory [179]. This would be useful as regards improving the visual notation of both methods, thus making them easier to understand and use.

7.2.7 Threats to validity for the family of the controlled experiments

We must consider certain issues which could threaten the validity of this experiment. With regard to its *internal validity*, the main threats are: learning effect, fatigue effects, participant experience, information exchange among participants, understandability of the documents, and instrumentation validity. The learning effect was mitigated by ensuring that each group of participants worked with the two methods, on two different experimental objects, using a within-participant experimental design. We mitigated the fatigue effects by executing the experiment in a time slot of 1 hour per session. Regarding the participants' experience, the random heterogeneity of subjects is always present when experimenting with students and we are also conscious that they had no previous knowledge of the value-driven methods being compared. Furthermore, if the knowledge of the students involved in the experiment could be assumed to be comparable to that of junior industry professionals, the working pressure and the overall environment in

industry is different. The experiment should be replicated with participants with experience in value-driven modeling. Nevertheless, the experience collected in this first study allows us to refine the material and tasks with the objective of performing a replication in an industrial setting. In order to minimize the information exchange among participants, they were monitored by the experimenters to avoid communication biases while performing the tasks. The understandability of the material was alleviated by performing a pilot study and making it available in three languages (Spanish, Portuguese-PT, and Portuguese-BR). Finally, the selection of different objects in the study may have affected the instrumentation validity and thus biased the results. We mitigated this threat by conducting a pilot experiment to assess both the complexity of the objects and to attempt to identify mistakes in the experimental material.

With regard to *external validity*, the main threats are: representativeness of the results, and the size and complexity of the tasks that might affect the generalization of the results. The representativeness of the results may be affected by the software systems used and the context of the participants selected. We mitigated the selection of software systems by considering a set of artifacts with a similar size and complexity, containing representative artifacts from an existing value-driven development method (i.e., e3value). The size and complexity of the tasks may also affect the external validity. We decided to use relatively small tasks since a controlled experiment requires that participants complete the assigned tasks in a limited amount of time. To confirm or contradict the achieved results, we plan to conduct case studies with larger and more complex tasks.

With regard to *construct validity*, the main threats are: the measures applied in the data analysis and the reliability of the questionnaire. We mitigated this by using measures that are commonly applied in other empirical-based software engineering works (including controlled experiments [3, 4, 23, 265] and meta-analysis [115, 165, 193, 249]). In particular, effectiveness was measured using an information retrieval based approach (see Section 7.2.2.1). The subjective variables are based on TAM [66, 147]. The reliability of the questionnaire was tested using the Cronbach test.

With regard to *conclusion validity*, the main threats are: the data collection and the validity of the statistical tests applied. In the case of the data collection, we applied the same data-extraction procedure in each individual experiment and ensured that each dependent variable was calculated with the same formula. With regard to the validity of the statistical tests proposed, we chose those that are most commonly employed in the empirical software engineering field (both for a simple experiment and for meta-analysis) owing to their robustness and sensitivity [173]. Finally, the meta-analysis results may be threatened by the reduced sample size. The effect size for each dataset was found to be small and moderate. To investigate this issue, we plan to conduct further experiment with a large number of participants.

7.3 Evaluating the methods RAMA and KAOS4Services

This section described an **online survey** to evaluate the perceived efficacy (that is, the *perceived ease of use* and the *perceived usefulness*) and the intention of use of the methods **RAMA** and KAOS4Services (the software layer of the framework.) It is important to note that we chose to do a single experiment to evaluate both methods rather than an experiment for each method because of the difficulty of recruiting participants. Therefore, to perform this evaluation, we created six research questions as follows:

RQ1. *Is the RAMA method perceived as easy to use?*

RQ2. *Is the RAMA method perceived as useful?*

RQ3. *Do the participants intend to use the RAMA method in the future?*

RQ4. *Is the KAOS4Services method perceived as easy to use?*

RQ5. *Is the KAOS4Services method perceived as useful?*

RQ6. *Do the participants intend to use the KAOS4Services method in the future?*

7.3.1 Experiment design

To evaluate the six research questions, we followed the **GQM** process [22] and defined the goal of this experiment to **analyze** the methods **RAMA** and KAOS4Services **for the purpose of** verifying the perceived efficacy of the methods **with respect to** the perceived ease of use and the perceived usefulness, **from the point of view of** software engineers, **in the context of** people with a background in computer science (with full or on-going courses) and who have participated in at least one information systems software development project in the industry.

Context of the experiment: We focused our evaluation on software engineers with a background in computer science (with full or on-going courses degrees) and who have participated in at least one information systems software development project in the industry. All the twenty-four participants were volunteers and were aware of the practical and pedagogical purposes of the survey, but did not know the experimental hypotheses nor did they have previous knowledge about **RAMA** or KAOS4Services.

Hypotheses formulation: We formulated twelve null hypotheses, defined in a one-tailed manner, as we want to analyze the perceived effect of our method on the variables. Each null hypothesis and its alternative are presented as follows:

H1-0: There is no significant perceived ease of use of the RAMA method

H1-a: The RAMA method is perceived as easy to use.

H2-0: There is no significant perceived usefulness of the RAMA method

H2-a: The RAMA method is perceived as useful.

H3-0: Participants do not have a significant intention to use the RAMA method

H3-a: Participants intend to use the RAMA method in the future.

H4-0: There is no significant perceived ease of use of the KAOS4Services method

H4-a: The KAOS4Services method is perceived as easy to use.

H5-0: There is no significant perceived usefulness of the KAOS4Services method

H5-a: The KAOS4Services method is perceived as useful.

H6-0: Participants do not have a significant intention to use the KAOS4Services method

H6-a: Participants intend to use the KAOS4Services method in the future.

Selected variables: The independent variable of interest is understanding the methods after watching some video classes. Hence, the experiment uses only one treatment: watch the video classes. The dependent variables are perception-based, assessing the participants' perceptions of methods. They are based on TAM [66]. The perceived efficacy [66] of the method can be broken down into three subjective dependent variables:

- **Perceived Ease of Use (PEOU):** refers to the degree to which a person believes that learning and using our method would not require significant effort.
- **Perceived Usefulness (PU):** refers to the degree to which a person believes that using our method will increase her/his job performance within an organizational context.
- **Intention to Use (ITU):** refers to the extent to which a person intends to use a particular method. It represents a perceptual judgment of the method's efficacy, that is, whether it is cost-effective and is commonly used to predict the likelihood of acceptance of a method in practice.

The three subjective variables were measured using a 5-point Likert scale questionnaire with a set of 12 closed-questions: 5 questions for perceived ease of use (PEOU), 5 for perceived usefulness (PU) and 2 for intention to use (ITU). They were formulated using the opposing statement format. So, each question contains two contradictory statements representing the max and min possible values (5 and 1), where 3 is considered a neutral perception. The aggregated value is the arithmetical mean of the answers to the questions associated with each perception-based variable. We used Cronbach's alpha test [184] to evaluate the reliability of the survey and of each variable.

Experiment design: Each participant should evaluate both methods. Because of this, we established two groups (each evaluating a different first method), and the participants were randomly assigned to each group. Table 7.6 summarizes the design of the survey. The comprehension of the video classes may also affect the application of the methods. We alleviated the influence of this factor by creating four short (the largest was 11 minutes) video classes (e.g., Introduction to [EBSE](#), [DVD](#) method, [RAMA](#) method, and [KAOS4Service](#) method).

Table 7.6: Experiment design

Phase	Groups		Is the experimental material different?
	A	B	
Part 1 (Introduction / levelling)	Introduction to EBSE and DVD method	Introduction to EBSE and DVD method	No
Part 2 (Classes evaluation)	Questionnaire	Questionnaire	No
Part 3 (1st method class)	KAOS4Services method	RAMA method	Yes
Part 4 (class evaluation)	Questionnaire	Questionnaire	No
Part 5 (1st method evaluation)	Questionnaire	Questionnaire	No
Part 6 (2nd method class)	RAMA method	KAOS4Services method	Yes
Part 7 (class evaluation)	Questionnaire	Questionnaire	No
Part 8 (2nd method evaluation)	Questionnaire	Questionnaire	No

Analysis procedure: We chose statistical tests for their robustness and sensitivity to analyze the data collected. As usual, in all the tests we decided to accept a probability of 5% of committing a Type-I-Error [265], rejecting the null hypothesis when it is true. The normality of the data distribution was tested with the Shapiro-Wilk test [223]. When there is only one sample to analyze, we used the T-test to one sample when data could assume the normal distribution to verify our hypotheses. However, when the data could not assume the normal distribution, then we applied the Wilcoxon-test. If there are independent samples, we applied the parametric one-tailed t-test for independent samples [138] when data could assume the normal distribution. However, when data did not assume the normal distribution, we applied the non-parametric Mann-Whitney test [61].

7.3.2 Discussion of the evaluation results for [RAMA](#) and [KAOS4Services](#)

With this survey, we aimed at analyzing the variables [PEOU](#), [PU](#), and [ITU](#) of the two methods forming the software layer of the framework. As the software layer of the Framework consists of the [RAMA](#) and [KAOS4Services](#) methods, there is a possibility of obtaining a positive final result for the software layer of the framework, but without getting a good result for the two methods. That is, it could be that the framework obtains a very good final result because one of the methods was considered excellent by the participants concerning [PEOU](#), [PU](#), and [ITU](#) in a way hiding the adverse effect of the other method. Because of this, initially, we analyzed the results of the methods individually to check for the hidden adverse effect.

What is the perceived efficacy and intention to use of the RAMA method? A 5-point Likert scale was used for the answers of the questions with 3 considered a neutral result (neither positive nor negative). A mean greater than 3 indicates a good result for the analyzed variable (e.g., **PEOU**, **ITU**, or **PU**), and a mean lower than 3 indicates a negative result. The averages obtained were: **PEOU**=4.5273 (with Std = 0.41194), **PU**=4.3977 (Std = 0.41303), and **ITU**=4.6818 (Std = 0.36337). As all the averages are higher than 3, the method obtained favorable results for all variables analyzed. However, for a definite answer, it is necessary to check if the difference between the means found is significantly higher than 3 (see H1-0, H2-0, and H3-0 hypotheses formulation in Section 7.3.1).

To analyze these hypotheses, we have to verify if the sample has a normal distribution through the Shapiro-Wilk test. The results are: **PEOU** = 0.060, **PU** = 0.039, and **ITU**= 0.000. As the p-values obtained for **PU** and **ITU** are less than 0.05, the samples for these variables have NOT a normal distribution and therefore we have to use the Wilcoxon non-parametric test for them (**PU** result = 0.001 and **ITU** result = 0.001). On the other hand, as the **PEOU** result is higher than 0.05, we can assume it has a normal distribution and therefore we should use the one-sample T-test (**PEOU** result = 0.001). As the p-values are less than 0.05, we can assume that the results are indeed significantly higher than 3, confirming all three alternative hypotheses (H1-a, H2-a, and H3-a). In conclusion, **RAMA** was perceived as easy to use and useful, and participants intend to use it in the future (if necessary).

What is the perceived efficacy and intention to use of the KAOS4Services method? The analysis performed for the **KAOS4Services** method was the same as for the **RAMA** method. In this way, we initially verified the means obtained for each variable, and obtained **PEOU**= 3.7455, **PU**= 3.7727, and **ITU**= 4.0455 with standard deviations equal to 0.48671, 0.51124, and 0.46057, respectively. Similarly to the **RAMA** method, all variables obtained an average higher than 3, that is, all are apparently positive. To confirm these positive results, we checked the hypotheses H4-0, H5-0, and H6-0 to verify if the difference between the averages found is significantly higher than 3 (see hypotheses formulation described in Section 7.3.1).

Then, we verified if the sample has a normal distribution through the Shapiro-Wilk test (**PEOU**= 0.425, **PU**= 0.000, and **ITU**= 0.004). As the p-values obtained for **PU** and **ITU** are less than 0.05, then the samples for these variables have NOT a normal distribution. Consequently, we used the Wilcoxon non-parametric test and obtained **PU**= 0.001 and **ITU**= 0.001. As the **PEOU** result is higher than 0.05, then it has a normal distribution and, therefore, we have to use the one-sample T-test (**PEOU**= 0.001).

As mentioned previously, p-values less than 0.05 mean that the results are significantly higher than 3. With all the alternative hypotheses were confirmed (H4-a, H5-a, and H6-a), we can conclude that **KAOS4Services** was perceived as easy to use and useful, and participants intend to use it in the future (if necessary).

7.3.3 Threats to validity

Before analyzing the results of the survey, we did some preliminary analysis to look for bias in the structure of the questionnaire or in the quality of the video classes given to the participants prior to answering the questions.

Questionnaire analysis: Cronbach's alpha was used to examine the reliability of the questionnaire, and the result obtained for the questionnaire was: [PEOU](#) questions is 0.855, [PU](#) questions is 0.722, [ITU](#) questions is 0.718, while that for the whole questionnaire is 0.825. This means that the questionnaire can be considered reliable (Cronbach's alpha is higher than 0.7 [184]).

Video classes analysis: We measured the quality of video classes using a 5-point Likert scale questionnaire with a set of 6 closed-questions: 3 for the video quality and 3 for the audio quality. Once more, the value 3 is considered to be a neutral result and results higher than 3 are considered positive. The descriptive analysis shows that the [KAOS4Services](#) video quality was 4.40 (Std 0.71), [RAMA](#)'s video quality was 4.7 (Std 0.33), and mean of videos quality was 4.57.

Regarding the audio quality, the results for the [KAOS4Services](#)' were 4.47 (Std 0.62), and for [RAMA](#)'s were 4.65 (Std 0.47), and the mean of the audios' quality was 4.56. Even though all the results were higher than 3, we must verify if the difference is significantly different from 3. For this, we verified if the dataset has a normal distribution using Shapiro-Wilk test, obtaining [KAOS4Services](#)' video quality= 0.001, [RAMA](#)'s video quality= 0.001, [KAOS4Services](#)' audio quality= 0.001, and [RAMA](#)'s audio quality= 0.001. These results show that [KAOS4Services](#)' and [RAMA](#)'s video and audio quality, has NOT a normal distribution (as $p\text{-value}<0.05$). Consequently, we used the non-parametric Wilcoxon test to verify the significance of the results ([KAOS4Services](#)' video quality= 0.001, [RAMA](#)'s video quality= 0.001, [KAOS4Services](#)' audio quality= 0.001, and [RAMA](#)'s audio quality= 0.001). As the results of Wilcoxon test were lower than 0.05, both audio and video qualities are considered good (significantly higher than 3).

7.4 Comparing RAMA and KAOS4Services

Even with positive results (for [PEOU](#), [PU](#), and [ITU](#)) for both methods, the fact is that these methods differ significantly in content. [RAMA](#) uses an agile development methodology while [KAOS4Services](#) uses a more traditional goal-oriented software development methodology. The final output created by the methods also differ. The architectural reference models created by [RAMA](#) and [KAOS4Services](#) change significantly due to the amount of different information obtained during the execution of the method used. Therefore, it is relevant to know which one was better evaluated by the participants, to identify which is the most promising method to be accepted in industry in the future and what

possible improvements could be worth making. In addition, we would like to know if the framework's software layer (both methods together) was perceived as easy to use and useful as well as if the participants intend to use it in the future. To achieve these aims, we extended the previous experiment as discussed next.

7.4.1 Experiment design

To perform this evaluation, we used the data already obtained in the previous experiment, adding to the previous six research questions another six, as follows:

RQ7. *Which of the methods is perceived as easier to use (PEOU), RAMA or KAOS4services?*

RQ8. *Which of the methods is perceived as most useful (PU), RAMA or KAOS4services?*

RQ9. *Which of the methods are the participants most likely to use in the future (ITU), RAMA or KAOS4Services?*

RQ10. *Is the framework's software layer perceived as easy to use (PEOU)?*

RQ11. *Is the framework's software layer perceived as useful (PU)?*

RQ12. *Do the participants intend to use the framework's software layer in the future (ITU)?*

Consequently, we created new hypotheses to each new research question. Each null hypothesis and its alternative are as follows:

H7-0: There is no significant difference between the perceived ease of use (PEOU) of RAMA and KAOS4Services

H7-a: The RAMA method is perceived as easier to use than the KAOS4Services method.

H8-0: There is no significant difference between the perceived usefulness of RAMA and KAOS4Services

H8-a: The RAMA method is perceived as more useful than the KAOS4Services method.

H9-0: There is no significant difference between the intention to use RAMA and KAOS4Services

H9-a: The RAMA method is perceived as more likely to be used than the KAOS4Services method.

H10-0: There is no significant perceived ease of use of the Framework's software layer

H10-a: The Framework's software layer is perceived as ease of use.

H11-0: There is no significant perceived usefulness of the Framework's software layer

H11-a: The Framework's software layer is perceived as useful.

H12-0: Participants do not have a significant intention to use the Framework's software layer

H12-a: Participants intend to use the Framework's software layer in the future.

(Disclaimer: although we have no reason to believe that one method is better than the other, the formulation of the H7-a, H8-a, and H9-a hypotheses starts with the [RAMA](#) method by chance, and we could have chosen the [KAOS4Services](#) method to start those formulations.)

Analysing the hypotheses H7, H8, and H9. We initially checked whether the data samples have a normal distribution using the Shapiro-Wilk test ($PEOU = 0.094$, $PU = 0.000$, and $ITU = 0.000$). As the p-values obtained for [PU](#) and [ITU](#) were less than 0.05, the samples for these variables have NOT a normal distribution, and therefore we used the Mann-Whitney non-parametric test for them ($PU = 0.000$ and $ITU = 0.000$). The [PEOU](#) result was higher than 0.05 (hence has a normal distribution), we used the one-tailed t-test for independent samples ($PEOU = 0.000$). P-values less than 0.05 mean that the results between the methods are significantly different, in other words, the results obtained by the [RAMA](#) method are significantly higher than those obtained by [KAOS4Services](#). Therefore, [RAMA](#) was perceived as easier to use and more useful than [KAOS4Services](#), confirming the hypothesis H7-a and H8-a. Also, participants claimed to have higher intention to use [RAMA](#) in the future (if necessary) than [KAOS4Services](#), confirming the H9-a.

Analysing hypotheses H10, H11, and H12. Given the results obtained by the [RAMA](#) and [KAOS4Services](#) methods, it seems reasonable to expect that [PEOU](#), [PU](#), and [ITU](#) will favor the software layer of the framework. To verify this result statistically, we calculated the value that each participant assigned to [PEOU](#), [PU](#) and [ITU](#) by the arithmetic mean of the amounts that were assigned to the [RAMA](#) and [KAOS4Services](#) methods. For example, if the [PEOU](#) result for [RAMA](#) was 5 and for [KAOS4Services](#) was 4, then the [PEOU](#) value for the Framework's software layer was 4.5 $[(5 + 4) / 2]$. Thus, the average of results considering all participants were $PEOU = 4.136$ (std = 0.3155), $PU = 4.0827$ (Std = 0.35341), and $ITU = 4.3636$ (Std = 0.32484). As the values are higher than 3, we need to check if the results are significantly higher than 3. To make this check, we must first examine whether the samples have a normal distribution through the Shapiro-Wilk test ($PEOU = 0.494$, $PU = 0.007$, and $ITU = 0.180$). As the values obtained for [PEOU](#) and [ITU](#) were higher than 0.05, then the samples for these variables have a normal distribution. Consequently, we used the parametric one-sample T-test for them ($PEOU = 0.000$ and $ITU = 0.000$). In contrast, since the result for [PU](#) was less than 0.05, then it does NOT have a normal distribution and, consequently, we used Wilcoxon's nonparametric test for it ($PU = 0.000$). The results of these tests show that the software layer of the Framework was perceived as easy to use and useful by the participants ([PEOU](#) and $PU < 0.05$), confirming

H10-a and H11-a. Besides, participants stated that they intend to use it in the future if they need to implement a value-based software development approach ($ITU < 0.05$), confirming H12-a.

7.4.2 Complementary analysis

To confirm that there was no bias regarding the order of which the methods were evaluated, we analyzed if the results obtained by group A who performed the experiment evaluating [KAOS4Services](#) and later [RAMA](#) (see Table 7.6) is NOT significantly different from group B that ran the experimental tasks in the reverse order. We also verified the size of the effect of the results found as we would have to increase our sampling if it was considered small.

Order of the survey: The absence of bias in the order of presentation of the methods, was checked by performing the Mann-Whitney tests (in [PU](#) and [ITU](#)) and one-tailed t-test for independent samples, grouping the results by group (group 1 = [KAOS4Services](#) > [RAMA](#) and group 2 = [RAMA](#) > [KAOS4Services](#)). The results were [PEOU](#) = 0.153, [PU](#) = 0.194, and [ITU](#) = 0.148, with p-values higher than 0.05. This means that the results between the groups are not significantly different, confirming that the order of execution of the survey did not have an impact on the results.

Effect size: The effect size is calculated with *Cohen's d* [59] and *effect size r*. (effect-size correlation) [184] using the means and standard deviations of two groups (treatment and control)⁴. Table 7.7 shows the result of these calculations per variable.

Table 7.7: Effect size of survey

Variable	<i>Cohen's d</i>	<i>effect size r</i>
PEOU	+1.733951811763858	0.6550639594530521
PU	+1.3448479625626903	0.5580034118320171
ITU	+1.533893328958535	0.6085714643625094

A positive value of *Cohen's d* (sign of +) indicates that the effect favours the [RAMA](#) method. In contrast, a negative value (sign of -) indicates that the effect favours the [KAOS4Services](#) method. Regarding the *effect size r*, Cohen suggests that a value between |.10| and |.29| represents a “small” effect size, between |.30| and |.49| represents a “moderate” effect size, and larger than |.5| represents a “large” effect size [59]. The results obtained for [PEOU](#), [PU](#), and [ITU](#) are higher than 0.5 (see Table 7.7), meaning that the results are statistically significant.

⁴Details on how to calculate *Cohen's d* and *effect size r* can be found in [29, 30, 59].

7.4.3 Discussion of the results

The descriptive statistics indicate that the **RAMA** method was perceived as easy to use ($\text{mean}_{\text{PEOU}} = 4.52$) and useful ($\text{mean}_{\text{PU}} = 4.39$), and the participants intend to use it in the future ($\text{mean}_{\text{ITU}} = 4.68$). These facts were confirmed through the parametric one-sample T-test, where the p-value for all variables was inferior to 0.05. One plausible justification for the perceived easy to use is that the **RAMA** method is lean and uses the tenth agile principle (e.g., simplicity)[9] and nine agile practices (e.g., user stories) effectively (see Section 5.3.1). The responses to the questionnaire's open questions indicated that *"the [RAMA] method is very simple, easy to learn and use. Despite having the step-by-step that can give a bureaucratic impression, I think it is viable in the agile development precisely because it is not bureaucratic"* or *"the [RAMA] method is very simple and easy to use"*. Also, three participants suggest building a support tool as a way to make the method easier to use. The combination of software architecture and agile development has received significant attention in recent years [267]. However, ways of combining them is still a challenging issue [2]. We believe that this challenge is being well addressed by the **RAMA** method, making it useful to the architects. Our assumption is highlighted by some answers to open questions from the questionnaire, for example, *"the [RAMA] method is very interesting. I think it is very useful to create software more aligned with the business in a very simple way"*. However, some improvement points to leave the most useful method were also reported, for example, *"to use more precise algorithms to identify overlapping concepts"* or *"to use more agile practices for planning (e.g., planning poker)"*. About the intend to use of the participants, the answers suggest that the main reason is the mix of easily and usefully of the **RAMA** method, for example, *"the [RAMA] method generates reference architecture models easily. These models are useful for discussion among the development team"* or *"I think the [RAMA] method is lean [very objective]. This makes it easy to use"*.

The descriptive statistics indicate that the **KAOS4Services** method was perceived as easy to use ($\text{mean}_{\text{PEOU}} = 3.74$) and useful ($\text{mean}_{\text{PU}} = 3.77$), and the participants intend to use it in the future ($\text{mean}_{\text{ITU}} = 4.04$). These facts were confirmed through the parametric one-sample T-test and the Wilcoxon non-parametric test, all with p-values inferior to 0.05. We believe that **KAOS4Services** is perceived as easy to use because its set of heuristics make it very systematic. Our assumption is supported by some of the answers to the questionnaire, for example, *"the [KAOS4Services] method is very simple and systematic"* or *"it is well bound and justified"*. Some participants suggest to automatize the heuristics or change the goal-oriented approach to make the method easier to use, for example, *"automate all heuristics"*, or *"heuristics should be transparent to users. I think the user does not need to memorize all these heuristics. Also, I do not like KAOS. If you change KAOS for something else, maybe the method will get easier."*, or yet *"explore BPMN instead of KAOS"*. **KAOS4Services** is perceived useful, because its output gives a good overview of the software that must be built. The open questions did not give us any support to justify our supposition because most participants did not write the reason why they think the method is useful. About

the intend to use of [KAOS4Services](#) in the future, some reasons were described by the participants, such as “*I would use the [KAOS4Service] method because I can see traceability of the business value in software services*”, “*[the KAOS4Services method] creates useful models to understand the system requirements*”, and yet “*the [KAOS4Services] method is well detailed and generates a good enough model for discussing on the structure of software*”.

Regarding the comparison between [RAMA](#) and [KAOS4Services](#) methods, [RAMA](#) was perceived as easier to use and more useful than [KAOS4Services](#). Also, participants claimed to have higher intention to use [RAMA](#) in the future (if necessary) than [KAOS4Services](#). The questionnaire’s open questions also show that the likelihood of intention to use the methods in the future is probably related to the easiness of using the method, as per answers like “*a thousand times this method [RAMA] to the other [KAOS4Services]. I would use this method [RAMA] because it is very simple and generates a useful reference model. It helps in the modularization of the system being developed*”.

Regarding the framework’s software layer, the analysis show that the software layer of the Framework was perceived as easy to use and useful by the participants. Besides, participants stated that they intend to use it in the future if they need to implement a value-based software development approach. These facts could not be different since the individual results of [RAMA](#) and [KAOS4Services](#) were positive for all three variables analyzed.

7.4.4 Threats to validity for the survey

With regard to its *internal validity*, the main threats are: learning effect, fatigue effects, participant experience, and understandability of the video classes. The learning effect was mitigated by ensuring that each group of participants watched video classes of the two methods using a within-participant experimental design. We mitigated the fatigue effects by creating short video classes (the longest lasted about 11 minutes). Regarding the participants’ experience, they had no previous knowledge of the value-driven methods being compared. The understandability of the video classes was alleviated by making them available in the native language of the participants (Portuguese-BR) and checking the quality of the audio and video through the questionnaire.

With regard to *construct validity*, the main threats are: the measures applied in the data analysis and the reliability of the questionnaire. We mitigated this by using measures that are commonly applied in other empirical-based software engineering works [3, 4, 23, 265]. In particular, the variables are based on [TAM](#) [66, 147]. The reliability of the questionnaire was tested using the Cronbach test.

With regard to *conclusion validity*, the main threats are: the data collection and the validity of the statistical tests applied. In the case of the data collection, we ensured that each dependent variable was calculated with the same formula. With regard to the validity of the statistical tests proposed, we chose those that are most commonly employed in the empirical software engineering field owing to their robustness and sensitivity [173].

7.5 Final considerations

A set of different experiments was used to evaluate our proposal. We performed at least one experiment per method (DVD, RAMA, and KAOS4Services), hence evaluating the whole proposal. First, we evaluated the *perceived ease of use and perceived usefulness* of the DVD method. After, we conducted a family of three experiments controlled to compare our DVD method with the well-known e3value (introduced in Chapter 2.3.2) with respect to their effectiveness, efficiency, perceived ease of use, perceived usefulness and intention to use, completing the evaluation of the framework's business layer. Then, we evaluated the *perceived ease of use and the perceived usefulness*, of the RAMA and KAOS4Services methods. We also evaluated the participants' intention to use the methods in the future (if necessary). Finally, we compared the results obtained for these two methods, to understand which one approach (agility versus use of goals) was better accepted. As a consequence, we checked that DVD, RAMA, and KAOS4Services were perceived as ease of use and useful by the participants of the experiments. Participants also affirm that they intend to use these methods in the future if necessary. These results support positively the approach we developed to answer the starting research question, *How to derive value-centred architectural models systematically?* discussed in Chapter 1.

CONCLUSIONS AND FUTURE WORK

The present research work has its roots in the challenge of deriving a software architecture model from an early requirements specification representing the value exchanges of a business. Although much work has been published addressing issues related to software architecture and requirements engineering, no systematic approaches exist to support the aligned construction of an architectural design with the business values that characterize organizations in their marketplace. Such construction process is difficult, requiring skilled and experienced architects. With a first degree in information systems and 10 years of experience in industry, the major problem I have encountered in my work was related to software architecture, more specifically, how to structure an information system to make it reflect the business needs. Additionally, in the companies I've worked for, the information systems were built based on business processes often not aligned with the companies' business values. These two problems were the motivations for this [Ph.D.](#) work. Therefore, the general goal of my research work was to explore systematic and rigorous means to derive a software architecture from early requirements specifications while considering their alignment with the business goals of an organization.

8.1 The quest and respective research questions

This [Ph.D.](#) research work began by surveying the existing body of knowledge in business modeling and software architecture. The methodology used for building such survey followed the best practices of the [EBSE](#) discipline. The results were a secondary study about the business modeling practices and a tertiary study to catalogue the empirical studies on software architecture aiming at aggregating the main findings reported in those studies to provide an overview of the consolidated state of the art. From the results of this last study, we identified the absence of a secondary study on software architecture

derivation methods, what led us to conduct a systematic mapping study on this particular topic. Both the tertiary study and the two systematic mapping studies highlighted several problems and challenges that we focus in this [Ph.D.](#) research, particularly: (i) the lack of support to build architectural models considering the business values of an organization, and (ii) the strong dependence on the architects' experience and intuition. The need to guide less experienced architects in the task of deriving a software architecture from a requirements specification that reflects the organization's business values, led to the following general research question:

How to derive value-centred architectural models systematically?

This research question was first subdivided into two sub-research questions to consider the derivation of software architecture models from early requirements specifications and also the alignment of the end result with the business values of an organization:

SRQ1. How to derive architectural models from early requirements specifications?

SRQ2. How to align software architecture models with the business values of an organization?

As our goal was also to guide young professionals throughout architectural design, the general research question led to two other sub-research questions to consider traceable and simple (easy to use and useful) models and methods:

SRQ3. How to produce simple models and explicit guidelines?

SRQ4. How to include traceability mechanisms in the architectural derivation process?

A synthesis of the empirical findings of these sub-questions, that together address this [Ph.D.](#) research question, are discussed next.

How to derive architectural models from early requirements specifications? We created a value-driven framework with three different methods to derive a software reference architecture from an early requirements specification. First, we identified, through a systematic mapping study, the complexity and lack of rigor in the existing approaches to represent business values and, as a consequence, we created [DVD](#), an early requirements specification method to represent business value exchanges. From a [DVD](#) model, intermediate requirements models (e.g., conceptual model or goal-oriented model) and software architecture models are derived using consecutive sets of model-to-model transformation techniques. Note that models are used intensively to capture and represent the knowledge acquired during the whole architectural derivation process. These models assist communication between stakeholders, and help analysts both understand complex problems and derive software architectural models using abstractions. Therefore, models are used not only as documentation artifacts, but also as central elements in the software engineering process. By using sophisticated [MDE](#) techniques (e.g., metamodels,

model transformations and model transformation languages), it was possible to obtain source generators to derive a software architecture model based on early value-driven requirements models using [RAMA](#) and [KAOS4Services](#). Thus, this [MDE](#) approach enables software engineers to work at a higher level of abstraction, providing techniques and tools to alleviate the difficulty of deriving software architecture models, which may benefit mostly the less experienced [IT](#) professionals.

How to align the software architecture models with the business values of the organization? We performed three systematic literature studies, following the best practices of [EBSE](#) [149], to build a survey of the existing knowledge on business value modeling and software architecture. From these studies, we identified the existing limitations of the current approaches that range from the construction of business values representations to the representation and alignment of those concept values in an architectural model. Besides the survey of the concepts involved, we also relate those concepts through conceptual mappings using model-driven development techniques. These mappings allowed us to align business value concepts to intermediate requirements models concepts (e.g., conceptual and goal-oriented models) and, from here, generate architectural model concepts. The methods [RAMA](#) and [KAOS4Services](#) support this goal by using model transformation languages to automate various steps of their processes. The proposed software architecture derivation methods are value-driven and based on the value exchanges specified in a [DVD](#) model. Finally, the feasibility of our proposal was confirmed by the development of a set of DSLs as proof of concepts tools.

How to produce simple models and explicit guidelines? By creating simple models we risk rendering them useless if they are unable to represent the needed concepts and relationships. So we were careful to analyze which essential concepts should be represented in both the value model and the requirements intermediary models (for example, we used only a subset of the [KAOS](#) concepts). Additionally, we followed the mindmaps philosophy, aiming at inheriting their well-recognized characteristics, such as simplicity and high degree of cognitiveness. In what concerns the guidelines, we were aware of the risk of making the architectural derivation process bureaucratic and ineffective. To avoid this, all the derivation steps were defined systematically with concise processes and guidelines were defined to complement the derivation instructions with good practices, even though these guidelines are not a prerequisite for executing the processes. In order to simplify the production of mappings, we automated them using model transformation languages, making them transparent (i.e. black boxes) for end users. A set of experiments confirmed that our proposal is perceived as easy to use and useful, and that the involved participants in the experiments showed interest in using our proposal in the future.

How to include traceability mechanisms in the architectural derivation process? We used **MDE** to facilitate the traceability of concepts from the specification of business values to a software reference architecture model. This forward (start to end) and backward (end to start) traceability is very important to build software aligned with business values. The underlying idea was to start from a high-level specification (a **DVD** model) and, by applying successive transformations, obtain lower-level specifications (a conceptual model or **KAOS** model, and a software reference architecture). These transformations between models support forward traceability, since from a source model concept (e.g., value exchange from a **DVD** model) we reach a target model concept (e.g., goal in a **KAOS** model). To ensure backward traceability, we used metadata during the transformation to save the source concept identifier in the generated target concept. Also, we were careful to use only hierarchical models, to be able to identify the parents' and children' concepts.

8.2 Contributions

The present research contributes to the software engineering field with the conception and specification of a Value-Driven Software Architecture Framework and associated methodological and evaluation approach. The resulting specific contributions are discussed next with respect to the four research questions.

Contributions from SRQ1. The resulting contributions from investigating *how to derive architectural models from early requirements specifications*, can be summarized as follows:

- **DVD method.** **DVD** is a cognitive early requirements method aimed at analyzing and representing business values exchanges. It offers an environment wherein stakeholders can share their economic views. This environment includes metamod-els, a **DSL**, transformations scripts, processes, concept mappings and guidelines. From a business model, goal-oriented models can be derived using model-driven techniques, one per value exchange. The resulting requirements specification is modularized from a business economic perspective, facilitating requirements pri-oritization. The DVD's environment includes metamodels, DSLs, transformations scripts, processes, mappings and guidelines.
- **Business value model.** A business model was created for business modelers with-out much information technology background. It contains the main concepts of-fered by other known business models (e.g., e3value, **REA**, and **BMO**) to represent business values. It allows stakeholders to share their economic viewpoints in a semi-structured mindmap model. It is composed of seven main concepts: main actor, environment actor, value exchange, who starts the value exchange, value port, value object, and value level agreement. These concepts and relationships are used by the **DVD** method to produce a **DVD** model.

- **RAMA method.** [RAMA](#) is a value-centric method to address the architecture-agility combination. [RAMA](#) uses business value modeling, model-driven and conceptual modeling techniques together with agile practices to produce a feasible approach to combine software architecture design and agile development during the creation of the information system's reference architecture aligned with the organization's business values. The [RAMA](#)'s environment includes metamodels, [DSLs](#), transformations scripts, processes, mappings and guidelines.
- **KAOS4Services method.** The [KAOS](#) for Services method is a systematic approach to model [SOA](#) applications from goal-oriented models. This is achieved with model-driven techniques and a set of guidelines applied to goal concepts. We analyzed nine goal-oriented modeling languages to identify and align the [DVD](#) business value concepts with goal-oriented concepts. From this analysis resulted that [KAOS](#) was the approach with the larger number of concepts related to [DVD](#) concepts. This, together with the fact that both [KAOS](#) has been one of the most well-cited goal-oriented approaches and it builds a model of the whole system, not just of part of it, were the reasons for choosing [KAOS](#) for our work. The KAOS4Services's environment includes metamodels, [DSLs](#), transformations scripts, processes, mappings and guidelines.

Contributions from SRQ2. The contributions from investigating *how to align software architecture models with the business values of an organization* are as follows:

- **Systematic literature mapping study on business modeling.** This systematic offers and analyzes a complete list of the business modeling methods published in the literature. We identified that most of the studies found were created to solve problems related to business strategy. The various studies share a set of common concepts (such as actor, value, and value exchange) aiming at facilitating business understanding and analysis to improve business strategies.
- **knowledge aggregation in software architecture.** The Software Architecture research community has accumulated a large body of knowledge to create and evolve new techniques, tools, processes, methods, and frameworks. So, we first performed a tertiary study of the existing secondary systematic studies to aggregate consolidated findings on software architecture. This tertiary study facilitates the work of researchers and practitioners in learning about the coverage and main results of existing work as well as identifying relatively unexplored niches of research that need further attention. We also conducted a mapping study (secondary study) to identify and analyze existing software architecture derivation methods. Thus, we confirmed the lack of approaches for deriving architectural models from early requirements specifications and that the building process is strongly based on the architects' experience, making it difficult for novices.

- **Conceptual mapping.** A set of conceptual mappings was defined to align the areas of business and IT. We have created a DVD model to specify the essential concepts of business value modeling, and aligned these concepts with requirements models concepts (e.g., from conceptual and KAOS models). The operationalization of this alignment was achieved using model-driven development techniques. Similarly, the requirements models concepts were also mapped to architectural model concepts, facilitating once more the alignment with the business values. Thus, MDD is used as the means of connection, or the glue, between the models at the various levels of abstraction.

Contributions from SRQ3. The contributions from searching *how to produce simple models and explicit guidelines and mappings* are:

- **Set of experiments.** We performed a quasi-experiment and confirmed that DVD was perceived as ease to use and useful. Also, we executed a family of three controlled experiments to compare the DVD method with the e3value method, with respect to their *effectiveness, efficiency, perceived ease of use, perceived usefulness and intention to use*. The experiment was initially performed at Universitat Politècnica de València (UPV) in Spain and then replicated at Universidade NOVA de Lisboa in Portugal and also at Universidade Federal de Pernambuco in Brazil with participants with different backgrounds. A meta-analysis was performed to aggregate the empirical findings obtained in each experiment. The results favoured DVD with respect to the e3value method. Finally, we performed a quasi-experiment to evaluate RAMA and KAOS4Services with respect to the participants' *perceived ease of use, perceived usefulness and intention to use*. Both methods were also perceived as ease to use and useful. This set of experiments contributes to the scientific community with a set of evidence that can be used for future research.
- **Set of guidelines.** We created a set of guidelines that guide less experienced architects in the use of the DVD, RAMA and KAOS4Services methods, facilitating the the various steps of the software architecture derivation processes.

Contributions from SRQ4. Finally, the main contribution when studying *how to include traceability mechanisms in the architectural derivation process* is:

- **Traceability support approach.** We created a simple traceability approach joining model transformation technique and metadata. In essence, a model transformation defines a set of relationships between a set of source and target models elements, implicitly defining a set of predecessor-successor relationships (forward traceability). Additionally, for backward traceability, we added metadata in the target element about the source element, allowing us to identify the source elements that originated the target element.

8.3 Published results

Throughout our [Ph.D.](#) research, we have been able to publish at least one article or conference paper per Chapter of our thesis. In total, we have eight scientific publications, of which two in journals and six in international conferences. One of the papers was awarded the **Best Paper** in a premier conference in Information Systems (the 12th European, Mediterranean and Middle Eastern Conference on Information Systems – EM-CIS’17). This paper proposes a systematic way to provide an alignment between value models and software requirements models. Next is a summary list of the published works (see Table [1.1](#) for further details).

- Deriving Architectural Models from Requirements Specifications: a Systematic Mapping Study [[245](#)] in Information and Software Technology Journal (Impact Factor: 2.921) by Eric Souza, Ana Moreira, and Miguel Goulão, 2019.
- Comparing Business Value Modeling Methods: A Family of Experiments [[243](#)] in Information and Software Technology Journal (Impact Factor: 2.921) by Eric Souza et al., 2018.
- Towards an Agile Reference Architecture Method for Information Systems [[244](#)] in Hawaii International Conference on System Sciences 2018 (CORE A) by Eric Souza, Ana Moreira, and Fernando Wanderley.
- Deriving Services using KAOS Models [[237](#)] in 33rd ACM/SIGAPP Symposium On Applied Computing (CORE B) by Eric Souza and Ana Moreira.
- An approach to align business and IT perspectives during the SOA services identification [[240](#)] in 17th International Conference on Computational Science and Its Applications (CORE C) by Eric Souza, Ana Moreira, and Cristiano de Faveri.
- Aligning business models with requirements models [[239](#)] in European Mediterranean & Middle Eastern Conference on Information Systems (CORE B) by Eric Souza, Ana Moreira, and João Araújo.
- Evaluating the efficacy of value-driven methods: a controlled experiment [[241](#)] in 26th International Conference on Information Systems Development (CORE A) by Eric Souza, Silvia Abrahão, Ana Moreira, Emilio Insfran, and João Araújo.
- Comparing Value-Driven Methods: an experiment design [[238](#)] in 2nd International Workshop on Human Factors in Modeling (HuFaMo’16@MODELS’16) by Eric Souza, Ana Moreira, Silvia Abrahão, João Araújo, and Emilio Insfran.

These papers have been published with other authors, but in all of them I am the first author because I have been the leader of the work for all of them.

8.4 Future work

A research work is never completed, and the more results one obtains, the more interesting questions and challenges one finds to explore. However, research grants and milestones in life impose constraints that oblige us to set limits on our tasks. The topic of this research continues to excite us to the point that we plan to keep the collaboration with the Automated Software Engineering research group of the NOVA LINCS research lab even after the [Ph.D.](#) defense, and we will look for more collaborations and research funds to pursue this topic for the years to come. The foreseen directions of work include:

Value-based planning, control and evaluation. Research on value-based planning and control covering principles and practices to control costs, schedule, and product planning, as well as further validations, are still required, particularly:

- Complement the business value perspective with the use of scenarios. This scenarios can describe the dynamic component of a business, using value stream in the ecosystem view.
- Explore the [DVD](#) in the context of the tech startups. We believe that [DVD](#) is ready to be used to offer a fast overview of a business scenario; it can help during planning and simple financial feasibility analysis.
- Analyze our framework using NIMSAD dimension [[132](#)].
- Perform a new empirical study to define more representative model iconography, improving effectiveness and usability.
- Evaluate the proposal in real projects, to test its robustness under stressful conditions, what may show points of improvement or points of adaptation that were not identified during the research.

Value-based requirements engineering. Value-based requirements engineering embodies principles and practices to identify stakeholders, identifying the value propositions and reconciling these with a mutually satisfactory set of objectives for the system. Thus, further research would be useful to:

- Explore other goal-oriented languages beyond [KAOS](#) during the reference architecture modeling in a traditional software development.
- Explore other requirements specification approaches, compare the different results of the use of each approach, and create a roadmap to suggest which approach is better under which circumstances.

Value-based architecture. Value-based architecture comprises further adjustments of the system objectives with possible architectural solutions. Further research to extend our proposal includes:

- Improve the conceptual overlaps algorithm (Levenshtein distance algorithm [171]) used by the [RAMA](#) method to identify synonymous words.
- Develop the traceability viewpoint where from a selected element at any stage of development, all related elements are showed in a network map.
- Build an end-user tool in the cloud, integrating all [Proof of concept \(PoC\)](#) tools in a unique webtool accessible to anyone.

Value-based design and development. Value-based design and development requires techniques to guarantee that the system's objectives and value considerations aligned with the business, are then inherited by the software design and development practices. Some required research topics include:

- Develop recommendations systems to suggest architectural styles that are more adequate to consider the elements represented in a software architecture reference model generated by [RAMA](#) and [KAOS4Services](#).
- Create a catalog of business architecturally significant requirements in order to link them with architectural solutions and system quality attributes.

BIBLIOGRAPHY

- [1] G. Abowd, R. Allen, and D. Garlan. “Using style to understand descriptions of software architecture.” In: *ACM SIGSOFT Software Engineering Notes*. Vol. 18. 5. ACM. 1993, pp. 9–20.
- [2] P. Abrahamsson, M. A. Babar, and P. Kruchten. “Agility and architecture: Can they coexist?” In: *IEEE Software* 27.2 (2010).
- [3] S. Abrahao, E. Insfran, J. A. Carsí, and M. Genero. “Evaluating requirements modeling methods based on user perceptions: A family of experiments.” English. In: *Information Sciences* 181.16 (2011), pp. 3356–3378. DOI: [10.1016/j.ins.2011.04.005](https://doi.org/10.1016/j.ins.2011.04.005). URL: <http://linkinghub.elsevier.com/retrieve/pii/S0020025511001733>.
- [4] S. Abrahao, C. Gravino, E. Insfran, G. Scanniello, and G. Tortora. “Assessing the Effectiveness of Sequence Diagrams in the Comprehension of Functional Requirements: Results from a Family of Five Experiments.” In: *IEEE Transactions on Software Engineering*, volume 39, issue 3 (2013), pp. 327–342. DOI: [10.1109/TSE.2012.27](https://doi.org/10.1109/TSE.2012.27). URL: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6193111>.
- [5] ACM. *ACM Digital Library*. Online accessed: 22/Sep/2017. 2017. URL: <https://dl.acm.org/>.
- [6] A. Ahmad, P. Jamshidi, and C. Pahl. *Protocol for Systematic Literature Review*. Tech. rep. Dublin - Ireland: Dublin City University, 2012.
- [7] M. M. Ahmed and S. Letchmunan. “A systematic literature review on challenges in service oriented software engineering.” In: *International Journal of Software Engineering and Its Applications* 9.6 (2015), pp. 173–186.
- [8] M. M. Al-Debei and D. Avison. “Developing a unified framework of the business model concept.” In: *European Journal of Information Systems* 19.3 (2010), pp. 359–376.
- [9] A. Alliance. *Subway Map to Agile Practices*. Online accessed: 22/Sep/2017. 2015. URL: <https://www.agilealliance.org/agile101/subway-map-to-agile-practices/>.

- [10] H. Almari and C. Boughton. "Questionnaire report on matter relating to software architecture evaluation." In: *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2014 15th IEEE/ACIS International Conference on*. IEEE. 2014, pp. 1–6.
- [11] M. Almorsy. "Adaptive, Model-based Cloud Computing Security Management." Doctoral dissertation. Swinburne University of Technology, 2014.
- [12] S. Ambler. *Agile modeling: effective practices for extreme programming and the unified process*. John Wiley & Sons, 2002.
- [13] D. Ameller, C. Ayala, J. Cabot, and X. Franch. "How do software architects consider non-functional requirements: An exploratory study." In: *Requirements Engineering Conference (RE), 2012 20th IEEE International*. IEEE. 2012, pp. 41–50.
- [14] D. Amyot, J. Horkoff, D. Gross, and G. Mussbacher. "A lightweight GRL profile for i* modeling." In: *International Conference on Conceptual Modeling*. Springer. 2009, pp. 254–264.
- [15] B. Andersson, M. Bergholtz, A. Edirisuriya, T. Ilayperuma, P. Johannesson, J. Gordijn, B. Grégoire, M. Schmitt, E. Dubois, S. Abels, et al. "Towards a reference ontology for business models." In: *Conceptual Modeling (ER 2006) (2006)*, pp. 482–496.
- [16] J. Andrade Almeida. "Model-driven design of distributed applications." Doctoral dissertation. Centre for Telematics and Information Technology University of Twente, 2006.
- [17] A. I. Anton. "Goal-based requirements analysis." In: *RE'96*. IEEE. 1996, pp. 136–144.
- [18] A. Arsanjani, L.-J. Zhang, M. Ellis, A. Allam, and K. Channabasavaiah. "S3: A service-oriented reference architecture." In: *IT professional* 9.3 (2007).
- [19] V. F. Avelino. "MERUSA: metodologia de especificação de requisitos de usabilidade e segurança orientada para arquitetura." Doctoral dissertation. Universidade de São Paulo, 2005.
- [20] L. G. Azevedo, F. Santoro, F. Baião, T. Diirr, A. Souza, J. F. de Souza, and H. P. Sousa. "A method for bridging the gap between business process models and services." In: *iSys-Revista Brasileira de Sistemas de Informação* 6.1 (2014), pp. 62–98.
- [21] F. Bachmann, L. Bass, M. Klein, and C. Shelton. "Designing software architectures to achieve quality attribute requirements." In: *IEE Proceedings-Software* 152.4 (2005), pp. 153–165.
- [22] V. R. Basili and H. D. Rombach. "The TAME project: Towards improvement-oriented software environments." In: *IEEE Transactions on software engineering* 14.6 (1988), pp. 758–773.

- [23] V. R. Basili, R. W. Selby, and D. H. Hutchens. "Experimentation in software engineering." In: *IEEE Transactions on software engineering* 7 (1986), pp. 733–743.
- [24] L. Bass. *Software architecture in practice*. Pearson Education India, 2007.
- [25] L. Bass, M. Klein, and F. Bachmann. "Quality attribute design primitives and the attribute driven design method." In: *International Workshop on Software Product-Family Engineering*. Springer. 2001, pp. 169–186.
- [26] L. Bass, P. Clements, and R. Kazman. *Software architecture in practice*. Addison-Wesley Professional, 2003.
- [27] K. Beck. *Test-driven development: by example*. Addison-Wesley Professional, 2003.
- [28] K. Beck and E. Gamma. *Extreme programming explained: embrace change*. addison-wesley professional, 2000.
- [29] L. A. Becker. *Effect Size Calculators*. University of Colorado Colorado Springs. Available on <https://www.uccs.edu/lbecker/>. 1999.
- [30] L. A. Becker. *Effect Size (ES)*. University of Colorado Colorado Springs. Available on <https://www.uccs.edu/lbecker/effect-size.html>. 2000.
- [31] A Berre. "Service oriented architecture modeling language (SoaML)-specification for the UML profile and metamodel for services (UPMS)." In: *Object Management Group (OMG)* (2008).
- [32] D. Bianchini, C. Cappiello, V. De Antonellis, and B. Pernici. "Service identification in interorganizational process design." In: *IEEE Transactions on Services Computing* 7.2 (2014), pp. 265–278.
- [33] S. Biffl, M. Ciolkowski, and F. Shull. "A family of experiments to investigate the influence of context on the effect of inspection techniques." In: *Proceedings of the Empirical Assessment in Software Engineering, IEE* (2002).
- [34] S. Biffl, A. Aurum, B. Boehm, H. Erdogmus, and P. Grünbacher. *Value-based software engineering*. Springer Science & Business Media, 2006.
- [35] B. Boehm and L. G. Huang. "Value-based software engineering: A case study." In: *Computer* 36.3 (2003), pp. 33–41.
- [36] B. W. Boehm. "Value-based software engineering: Overview and agenda." In: *Value-based software engineering*. Springer, 2006, pp. 3–14.
- [37] B. W. Boehm and A. Jain. "An initial theory of value-based software engineering." In: *Value-Based Software Engineering*. Springer, 2006, pp. 15–37.
- [38] D. Bollier. *The Future of e-Commerce*. The Aspen Institute, 1996.
- [39] D. Bombonatti, M. Goulão, and A. Moreira. "Synergies and tradeoffs in software reuse—a systematic mapping study." In: *Software: practice and experience* 47.7 (2017), pp. 943–957.

- [40] A. Borg, A. Yong, P. Carlshamre, and K. Sandahl. "The bad conscience of requirements engineering: an investigation in real-world treatment of non-functional requirements." In: (2003).
- [41] A. Borgida, N. Ernst, I. J. Jureta, A. Lapouchnian, S. Liaskos, and J. Mylopoulos. "Techne: A (nother) requirements modeling language." In: *Computer Systems Research Group. Toronto, Canada: University of Toronto* (2009).
- [42] J. Bosch. *Design and use of software architectures: adopting and evolving a product-line approach*. Pearson Education, 2000.
- [43] J. Bosch. "Software architecture: The next step." In: *EWSA 3047* (2004), pp. 194–199.
- [44] M. Brambilla, J. Cabot, and M. Wimmer. "Model-driven software engineering in practice." In: *Synthesis Lectures on Software Engineering* 1.1 (2012), pp. 1–182.
- [45] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil. "Lessons from applying the systematic literature review process within the software engineering domain." In: *Journal of systems and software* 80.4 (2007), pp. 571–583.
- [46] L. C. Briand, Y Labiche, M Di Penta, and H Yan-Bondoc. "An experimental investigation of formality in UML-based development." In: *IEEE Transactions on Software Engineering*, volume 31, issue 10 (2005), pp. 833–849. DOI: [10.1109/TSE.2005.105](https://doi.org/10.1109/TSE.2005.105).
- [47] F. Budinsky, D. Steinberg, R. Ellersick, T. J. Grose, and E. Merks. *Eclipse modeling framework: a developer's guide*. Addison-Wesley Professional, 2004.
- [48] R. J. Buhr and R. S. Casselman. *Use case maps for object-oriented systems*. Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1995.
- [49] S. A. Butler. "Security attribute evaluation method: a cost-benefit approach." In: *Proceedings of the 24th international conference on Software engineering*. ACM. 2002, pp. 232–240.
- [50] J. N. Buxton and B. Randell. *Software Engineering Techniques: Report on a Conference Sponsored by the NATO Science Committee*. NATO Science Committee; available from Scientific Affairs Division, NATO, 1970.
- [51] T. Buzan and B. Buzan. "The Mind Map Book: How to Use Radiant Thinking to Maximize Your Brain's Untapped Potential." In: (1996).
- [52] I. T. Cameron and K. Hangos. *Process modelling and model analysis*. Vol. 4. Academic Press, 2001.
- [53] S. Casteleyn, F. Daniel, P. Dolog, and M. Matera. *Engineering web applications*. Vol. 30. Springer, 2009.
- [54] C. Catal and M. Atalay. "A Systematic Mapping Study on Architectural Analysis." In: *Information Technology: New Generations (ITNG), 2013 Tenth International Conference on*. IEEE. 2013, pp. 661–664.

- [55] R. Chitchyan, A. Rashid, P. Rayson, and R. Waters. "Semantics-based composition for aspect-oriented requirements engineering." In: *Proceedings of the 6th international conference on Aspect-oriented software development*. ACM. 2007, pp. 36–48.
- [56] L. Chung, B. A. Nixon, E. S.-K. Yu, and J. Mylopoulos. *Non-Functional Requirements in Software Engineering*. Boston, MA, 1999. ISBN: 0-7923-8666-3. DOI: [10.1007/978-1-4615-5269-7](https://doi.org/10.1007/978-1-4615-5269-7).
- [57] L. Chung, S. Supakkul, N. Subramanian, J. L. Garrido, M. Noguera, M. V. Hurtado, M. L. Rodríguez, and K. Benghazi. "Goal-oriented software architecting." In: *Relating Software Requirements and Architectures*. Springer, 2011, pp. 91–109.
- [58] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos. *Non-functional requirements in software engineering*. Vol. 5. Springer Science & Business Media, 2012.
- [59] J. Cohen. *Statistical power analysis for the behavioral sciences 2nd edn*. 1988.
- [60] W. R. Collins, K. W. Miller, B. J. Spielman, and P. Wherry. "How good is good enough?: an ethical analysis of software construction and use." In: *Communications of the ACM* 37.1 (1994), pp. 81–91.
- [61] W. J. Conover. *Practical Nonparametric Statistics*. 3rd. Wiley, 2006. ISBN: 8126507756, 9788126507757.
- [62] L. G. Cretu and F. Dumitriu. *Model-Driven Engineering of Information Systems: Principles, Techniques, and Practice*. CRC Press, 2014.
- [63] E. Dafikpaku, M. Eng, and M. Mcmi. "The strategic implications of enterprise risk management: A framework." In: *ERM Symposium*. Vol. 48. 2011.
- [64] L. Dai and K. Cooper. "Modeling and analysis of non-functional requirements as aspects in a UML based architecture design." In: *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, 2005 and First ACIS International Workshop on Self-Assembling Wireless Networks. SNPD/SAWN 2005. Sixth International Conference on*. IEEE. 2005, pp. 178–183.
- [65] A. Dardenne, A. Van Lamsweerde, and S. Fickas. "Goal-directed requirements acquisition." In: *Science of computer programming* 20.1-2 (1993), pp. 3–50.
- [66] F. D. Davis. "Perceived usefulness, perceived ease of use, and user acceptance of information technology." In: *MIS quarterly* (1989), pp. 319–340.
- [67] T. DeMarco and T. Lister. *Waltzing with bears: Managing risk on software projects*. Addison-Wesley, 2013.
- [68] W. M. van Der Aalst, A. H. Ter Hofstede, B. Kiepuszewski, and A. P. Barros. "Work-flow patterns." In: *Distributed and parallel databases* 14.1 (2003), pp. 5–51.
- [69] Z. Derzsi and J. Gordijn. "A Framework for Business/IT Alignment in Networked Value Constellations." In: *BUSITAL*. 2006.

- [70] E. Di Nitto and D. Rosenblum. “Exploiting ADLs to specify architectural styles induced by middleware infrastructures.” In: *Proceedings of the 21st international conference on Software engineering*. ACM. 1999, pp. 13–22.
- [71] J. L. Dietz. *Enterprise ontology: theory and methodology*. Springer Science & Business Media, 2006.
- [72] T. Diirr, L. G. Azevedo, F. Faria, F. Santoro, and F. Baião. “Utilizando SoaML para Modelagem e Geração de Código de Serviços em uma Abordagem SOA.” In: *Cadernos do IME-Série Informática* 30 (2013), pp. 38–49.
- [73] E. W. Dijkstra. “On the role of scientific thought.” In: *Selected writings on computing: a personal perspective*. Springer, 1982, pp. 60–66.
- [74] Dimitrios S. Kolovos, Louis Rose, Richard Paige, and Antonio García-Domínguez. *The Epsilon book*. 2018. URL: www.eclipse.org/epsilon/doc/book/.
- [75] S. Direct. *Science Direct Digital Library*. Online accessed: 22/Sep/2017. 2017. URL: <http://sciencedirect.com>.
- [76] Drichards tool. *Drichards tool*. Online accessed: 22/Jun/2018. 2018. URL: <https://www.mindmaps.app>.
- [77] DTSam. “An Overview of SoaML.” In: *Sparx Systems Enterprise Architect* (2011). <https://goo.gl/WsLJ5g>. Online accessed: 22/Sep/2018.
- [78] T. Dyba, B. A. Kitchenham, and M. Jorgensen. “Evidence-based software engineering for practitioners.” In: *IEEE software* 22.1 (2005), pp. 58–65.
- [79] Eclipse Foundation. *EuGENia GMF Tutorial*. Online accessed: 27/Aug/2018. URL: <https://goo.gl/Hx27Nu>.
- [80] Eclipse Foundation. *Sirius*. Online accessed: 27/Aug/2018. URL: <http://www.eclipse.org/sirius/>.
- [81] B. Elvesæter, A.-J. Berre, and A. Sadovykh. “Specifying Services using the Service Oriented Architecture Modeling Language (SoaML)-A Baseline for Specification of Cloud-based Services.” In: *CLOSER*. 2011, pp. 276–285.
- [82] Y. Eric. “Social Modeling and i*.” In: *Conceptual Modeling: Foundations and Applications*. Springer, 2009, pp. 99–121.
- [83] J. Erickson, K. Lyytinen, and K. Siau. “Agile modeling, agile software development, and extreme programming: the state of research.” In: *Journal of database Management* 16.4 (2005), p. 88.
- [84] T. Erl. *Service-oriented architecture: concepts, technology, and design*. Pearson Education India, 2005.
- [85] T. Erl. *Soa: principles of service design*. Vol. 1. Prentice Hall Upper Saddle River, 2008.

- [86] E. Evans. *Domain-driven design: tackling complexity in the heart of software*. Addison-Wesley Professional, 2004.
- [87] S. C.P. F. Fabbri, K. R. Felizardo, F. C. Ferrari, E. C. M. Hernandez, F. R. Octaviano, E. Y. Nakagawa, and J. C. Maldonado. “Externalising tacit knowledge of the systematic review process.” In: *IET software* 7.6 (2013), pp. 298–307.
- [88] D. Falessi, G. Cantone, and P. Kruchten. “Do architecture design methods meet architects’ needs?” In: *Software Architecture, 2007. WICSA’07. The Working IEEE/I-FIP Conference on*. IEEE. 2007, pp. 5–5.
- [89] D. Falessi, G. Cantone, R. Kazman, and P. Kruchten. “Decision-making techniques for software architecture design: A comparative survey.” In: *ACM Computing Surveys (CSUR)* 43.4 (2011), p. 33.
- [90] S. Faulk, R. Harmon, and D. Raffo. “Value-base software engineering: A value-driven approach to product-line engineering. 1st Int.” In: *Confon Software Product-Line Engineering, Colorado*. 2000.
- [91] R. T. Fielding and R. N. Taylor. *Architectural styles and the design of network-based software architectures*. Vol. 7. University of California, Irvine Doctoral dissertation, 2000.
- [92] E. Foundation. *Acceleo Query Language: Query and navigate in EMF models*. On-line accessed: 22/Jun/2018. URL: <https://www.eclipse.org/acceleo/documentation/>.
- [93] M. Fowler and J. Highsmith. “The agile manifesto.” In: *Software Development* 9.8 (2001), pp. 28–35.
- [94] W. B. Frakes and R. Baeza-Yates. *Information Retrieval: Data Structures & Algorithms*. New Jersey, USA: Prentice-Hall, Inc., Upper Saddle River, 1992.
- [95] S. Freudenberg and H. Sharp. “The top 10 burning research questions from practitioners.” In: *Ieee Software* 27.5 (2010), pp. 8–9.
- [96] M. Galster, A. Eberlein, and M. Moussavi. “Transition from requirements to architecture: A review and future perspective.” In: *Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2006. SNPD 2006. Seventh ACIS International Conference on*. IEEE. 2006, pp. 9–16.
- [97] M. Galster, M. Mirakhorli, J. Cleland-Huang, X. Franch, J. E. Burge, R. Roshandel, and P. Avgeriou. “Towards bridging the twin peaks of requirements and architecture.” In: *ACM SIGSOFT Software Engineering Notes* 39.5 (2014), pp. 30–31.
- [98] E. Gamma. *Design patterns: elements of reusable object-oriented software*. Pearson Education India, 1995.
- [99] D. Garlan. “Software architecture: a roadmap.” In: *Proceedings of the Conference on the Future of Software Engineering*. ACM. 2000, pp. 91–101.

- [100] P. Gerrard and N. Thompson. *Risk-based e-business testing*. Artech House, 2002.
- [101] J. Gonzalez-Huerta, E. Insfran, S. Abrahão, and G. Scanniello. “Validating a model-driven software architecture evaluation and improvement method: A family of experiments.” In: *Information and Software Technology* 57 (2015), pp. 405–429.
- [102] Google. *Google Scholar*. Online accessed: 22/Sep/2017. 2017. URL: <https://scholar.google.com>.
- [103] J. Gordijn. “E3-value in a Nutshell.” In: *Proceedings of International Workshop on E-business Modeling, Lausanne*. 2002.
- [104] J. Gordijn. “Value-based Requirements Engineering.” Doctoral dissertation. Vrije Universiteit Amsterdam, 2002.
- [105] J. Gordijn and J. Akkermans. “Value-based requirements engineering: exploring innovative e-commerce ideas.” In: *Requirements engineering* 8.2 (2003), pp. 114–134.
- [106] J. Gordijn, H. Akkermans, and H. Van Vliet. “Business modelling is not process modelling.” In: *Conceptual modeling for e-business and the web* (2000), pp. 40–51.
- [107] J. Gordijn, H. Akkermans, and H. Van Vliet. “What’s in an electronic business model?” In: *Knowledge engineering and knowledge management methods, models, and tools* (2000), pp. 1–26.
- [108] J. Gordijn, H. Akkermans, and J Van Vliet. “Designing and evaluating e-business models.” In: *IEEE intelligent Systems* 16.4 (2001), pp. 11–17.
- [109] R. C. Gronback. *Eclipse modeling project: a domain-specific language (DSL) toolkit*. Pearson Education, 2009.
- [110] P. Grünbacher, A. Egyed, and N. Medvidovic. “Reconciling software requirements and architectures with intermediate models.” In: *Software and Systems Modeling* 3.3 (2004), pp. 235–253.
- [111] P. Grünbacher, S. Köszegi, and S. Biffl. “Stakeholder value proposition elicitation and reconciliation.” In: *Value-Based Software Engineering*. Springer, 2006, pp. 133–154.
- [112] Q. Gu and P. Lago. “A stakeholder-driven service life cycle model for SOA.” In: *2nd international workshop on Service oriented software engineering: in conjunction with the 6th ESEC/FSE joint meeting*. ACM. 2007, pp. 1–7.
- [113] Q. Gu and P. Lago. “Exploring service-oriented system engineering challenges: a systematic literature review.” In: *Service Oriented Computing and Applications* 3.3 (2009), pp. 171–188.
- [114] E. Hans-Erik and P. Magnus. *Business Modeling with UML: Business Patterns at Work*. 2000.

-
- [115] W. Hayes. "Research synthesis in software engineering: a case for meta-analysis." In: *Software Metrics Symposium, 1999. Proceedings. Sixth International*. IEEE. 1999, pp. 143–151.
- [116] L. V. Hedges and I. Olkin. *Statistical Methods for Meta-Analysis*. 1st. Academia Press, 1985.
- [117] U. van Heesch and P. Avgeriou. "Mature architecting-a survey about the reasoning process of professional architects." In: *Software Architecture (WICSA), 2011 9th Working IEEE/IFIP Conference on*. IEEE. 2011, pp. 260–269.
- [118] J. E. Hirsch. "An index to quantify an individual's scientific research output." In: *Proceedings of the National academy of Sciences of the United States of America* 102.46 (2005), p. 16569.
- [119] C. Hofmeister, R. Nord, and D. Soni. *Applied software architecture*. Addison-Wesley Professional, 2000.
- [120] L. Hohmann. *Beyond software architecture: creating and sustaining winning solutions*. Addison-Wesley Longman Publishing Co., Inc., 2003.
- [121] M. B. Holbrook. *Consumer value: a framework for analysis and research*. Psychology Press, 1999.
- [122] C Huemer, A Schmidt, H Werthner, and M Zapletal. "'A UML Profile for the e3-Value e-Business Model Ontology"; Vortrag: Third International Workshop on Business/IT Alignment and Interoperability (BUSITAL'08) held in conjunction with CAiSE'08 Conference, Montpellier, Frankreich; 16.06. 2008; in:"Proceedings of the Third International Workshop on Business/IT Alignment and Interoperability (BUSITAL'08) held in conjunction with CAiSE'08 Conference", CEUR-WS, Vol-336 (2008), ISSN: 1613-0073; Paper-Nr. 1, 15 S." In: ().
- [123] IBM Analytics. *IBM SPSS Software: Deliver greater business results with Predictive Intelligence*. Available on <https://goo.gl/kfth4N>. 2018.
- [124] IEEE. *IEEE Xplore Digital Library*. Online accessed: 22/Sep/2017. 2017. URL: <https://ieeexplore.ieee.org>.
- [125] E. Insfran, S. Abrahão, J. González-Huerta, J. D. McGregor, and I. Ramos. "A multimodeling approach for quality-driven architecture derivation." In: *Building Sustainable Information Systems*. Springer, 2013, pp. 205–218.
- [126] B. ISO. "IEC 2382-1 1993." In: *Information technology. Vocabulary. Fundamental terms*. British Standards Institution (1994).
- [127] ISO/IEC/IEEE 42010. "International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers(IEEE): Systems and software engineering - Architecture description (ISO/IEC/IEEE 42010)." In: (2011).

- [128] S. Jalali and C. Wohlin. "Systematic literature studies: database searches vs. backward snowballing." In: *the ACM-IEEE international symposium*. New York, New York, USA: ACM Press, 2012, pp. 29–10. ISBN: 9781450310567. DOI: [10.1145/2372251.2372257](https://doi.org/10.1145/2372251.2372257).
- [129] P. Jamshidi, M. Sharifi, and S. Mansour. "To establish enterprise service model from enterprise business model." In: *Services Computing, 2008. SCC'08. IEEE International Conference on*. Vol. 1. IEEE. 2008, pp. 93–100.
- [130] P. Jamshidi, M. Ghafari, A. Ahmad, and C. Pahl. "A framework for classifying and comparing architecture-centric software evolution research." In: *Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on*. IEEE. 2013, pp. 305–314.
- [131] A. Jansen and J. Bosch. "Software architecture as a set of architectural design decisions." In: *Software Architecture, 2005. WICSA 2005. 5th Working IEEE/IFIP Conference on*. IEEE. 2005, pp. 109–120.
- [132] N. Jayaratna. *Understanding and Evaluating Methodologies: NIMSAD, a Systematic Framework*. New York, NY, USA: McGraw-Hill, Inc., 1994.
- [133] M. Jazayeri, A. Ran, and F. Van Der Linden. *Software architecture for product families: principles and practice*. Vol. 9. Addison-Wesley Reading, 2000.
- [134] Jennifer Horkoff and Eric Yu. *I-Star Wiki*. <http://istar.rwth-aachen.de>. [Online; accessed 03-May-2017]. 2017.
- [135] R. E. Johnson. "Components, frameworks, patterns." In: *ACM SIGSOFT Software Engineering Notes*. Vol. 22. 3. ACM. 1997, pp. 10–17.
- [136] N. M. Josuttis. *SOA in practice: the art of distributed system design*. "O'Reilly Media, Inc.", 2007.
- [137] F. Jouault, F. Allilaire, J. Bézivin, and I. Kurtev. "ATL: A model transformation tool." In: *Science of computer programming* 72.1-2 (2008), pp. 31–39.
- [138] N. Juristo and A. M. Moreno. *Basics of Software Engineering Experimentation*. 1st. Springer US, 2010. URL: <http://dl.acm.org/citation.cfm?id=1965114>.
- [139] J. Karlsson and K. Ryan. "A cost-value approach for prioritizing requirements." In: *IEEE software* 14.5 (1997), pp. 67–74.
- [140] J. Karlsson and K. Ryan. "Prioritizing requirements using a cost-value approach." In: *IEEE Software* 14.5 (1997), pp. 67–74.
- [141] V. Kartseva, J. Hulstijn, Z. Baida, J. Gordijn, and Y.-H. Tan. "Towards control patterns for smart business networks." In: *Proceedings of the Smart Business Networks Initiative Discovery Session*. Springer, Heidelberg (2006).
- [142] R. Kazman, L. Bass, G. Abowd, and M. Webb. "SAAM: A method for analyzing the properties of software architectures." In: *Software Engineering, 1994. Proceedings. ICSE-16., 16th International Conference on*. IEEE. 1994, pp. 81–90.

-
- [143] R. Kazman, M. Klein, and P. Clements. *ATAM: Method for architecture evaluation*. Tech. rep. Carnegie-Mellon Univ Pittsburgh PA Software Engineering Inst, 2000.
- [144] R. Kazman, J. Asundi, and M. Klein. “Quantifying the costs and benefits of architectural decisions.” In: *Software Engineering, 2001. ICSE 2001. Proceedings of the 23rd International Conference on*. IEEE. 2001, pp. 297–306.
- [145] S. Kelly and J.-P. Tolvanen. *Domain-specific modeling: enabling full code generation*. John Wiley & Sons, 2008.
- [146] T. Kelly and R. Weaver. “The goal structuring notation—a safety argument notation.” In: *Dependable systems and networks workshop on assurance cases*. 2004.
- [147] W. R. King and J. He. “A meta-analysis of the technology acceptance model.” In: *Information & Management, volume 46, issue 6* (2006), pp. 740–755. DOI: [10.1016/j.im.2006.05.003](https://doi.org/10.1016/j.im.2006.05.003). URL: <http://linkinghub.elsevier.com/retrieve/pii/S0378720606000528>.
- [148] B. Kitchenham and S. Charters. “Guidelines for performing systematic literature reviews in software engineering.” In: *Technical report, Ver. 2.3 EBSE Technical Report*. EBSE. sn, 2007.
- [149] B. Kitchenham, E. Mendes, and G. H. Travassos. “Cross versus within-company cost estimation studies: A systematic review.” In: *IEEE Transactions on Software Engineering* 33.5 (2007).
- [150] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman. “Systematic literature reviews in software engineering—a systematic literature review.” In: *Information and software technology* 51.1 (2009), pp. 7–15.
- [151] B. A. Kitchenham, T. Dyba, and M. Jorgensen. “Evidence-based software engineering.” In: *Proceedings of the 26th international conference on software engineering*. IEEE Computer Society. 2004, pp. 273–281.
- [152] B. A. Kitchenham, D. Budgen, and O. P. Brereton. “Using mapping studies as the basis for further research—a participant-observer case study.” In: *Information and Software Technology* 53.6 (2011), pp. 638–651.
- [153] A. G. Kleppe, J. B. Warmer, and W. Bast. *MDA explained: the model driven architecture: practice and promise*. Addison-Wesley Professional, 2003.
- [154] T. Kohlborn, A. Korthaus, T. Chan, and M. Rosemann. “Identification and analysis of business and software services—a consolidated approach.” In: *IEEE Transactions on Services Computing* 2.1 (2009), pp. 50–64.
- [155] D. S. Kolovos, N. Matragkas, J. R. Williams, and R. F. Paige. “Model Driven Grant Proposal Engineering.” In: *International Conference on Model Driven Engineering Languages and Systems - MODELS’14*. Springer. 2014, pp. 420–432.
- [156] L. Koskela. *Test driven: practical TDD and acceptance TDD for java developers*. Manning Publications Co., 2007.

- [157] A. Koski and T. Mikkonen. “What We Say We Want and What We Really Need: Experiences on the Barriers to Communicate Information System Needs.” In: *Requirements Engineering for Service and Cloud Computing*. Springer, 2017, pp. 3–21.
- [158] P. Kruchten. “An ontology of architectural design decisions in software intensive systems.” In: *2nd Groningen workshop on software variability*. Groningen, The Netherlands. 2004, pp. 54–61.
- [159] P. Kruchten. *The rational unified process: an introduction*. Addison-Wesley Professional, 2004.
- [160] P. Kruchten, H. Obbink, and J. Stafford. “The Past, Present, and Future of Software Architecture.” In: *IEEE Software* 23.2 (2006), pp. 22–30. DOI: [10.1109/MS.2006.59](https://doi.org/10.1109/MS.2006.59).
- [161] P. B. Kruchten. “The 4 + 1 view model of architecture.” In: *IEEE software* 12.6 (1995), pp. 42–50.
- [162] D. Kundisch and T. John. “Business model representation incorporating real options: an extension of e3-value.” In: *System Science (HICSS), 2012 45th Hawaii International Conference on*. IEEE. 2012, pp. 4456–4465.
- [163] T. Kühne. “Matters of (Meta-)Modeling.” In: *Software and System Modeling* 5.4 (2006), pp. 369–385.
- [164] V. I. Levenshtein. “Binary codes capable of correcting deletions, insertions, and reversals.” In: *Soviet physics doklady*. Vol. 10. 8. 1966, pp. 707–710.
- [165] M. W. Lipsey and D. B. Wilson. *Practical meta-analysis*. Sage Publications, Inc, 2001.
- [166] P. Loucopoulos and R. Zicari. *Conceptual modeling, databases, and CASE: an integrated view of information systems development*. John Wiley & Sons, Inc., 1992.
- [167] P. Loucopoulos, V. Kavakli, N. Prekas, C. Rolland, G. Grosz, and S. Nurcan. “Using the EKD approach: the modelling component.” In: (1997).
- [168] Z. Mahmood. “The promise and limitations of service oriented architecture.” In: *International journal of Computers* 1.3 (2007), pp. 74–78.
- [169] I. Malavolta, P. Lago, H. Muccini, P. Pelliccione, and A. Tang. “What industry needs from architectural languages: A survey.” In: *IEEE Transactions on Software Engineering* 39.6 (2013), pp. 869–891.
- [170] R. Malveau and T. J. Mowbray. *Software Architect Bootcamp*. Prentice Hall Professional Technical Reference, 2003.
- [171] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. vol. 1., Cambridge University Press, 2008.

-
- [172] S. Mary and G. David. "Software Architecture: Perspectives on an Emerging Discipline." In: *Prentice-Hall* (1996).
- [173] K Maxwell. *Applied statistics for software managers*. Prentice Hall. 2002.
- [174] W. E. McCarthy. "The REA accounting model: A generalized framework for accounting systems in a shared data environment." In: *Accounting Review* (1982), pp. 554–578.
- [175] J. Medeiros, A. Vasconcelos, M. Goulão, C. Silva, and J. Araújo. "An approach based on design practices to specify requirements in agile projects." In: *Proceedings of the Symposium on Applied Computing*. ACM. 2017, pp. 1114–1121.
- [176] N. Medvidovic, E. M. Dashofy, and R. N. Taylor. "Moving architectural description from under the technology lamppost." In: *Information and Software Technology* 49.1 (2007), pp. 12–31.
- [177] S. J. Mellor, K. Scott, A. Uhl, and D. Weise. *MDA distilled: principles of model-driven architecture*. Addison-Wesley Professional, 2004.
- [178] ModelMaker Tools BV. *SimpleMind Lite tool*. Online accessed: 22/Jun/2018. 2018. URL: <https://itunes.apple.com/br/app/simplemind-lite-mind-mapping/id439654198?mt=12>.
- [179] D Moody. "The Physics of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering." In: *IEEE Transactions on Software Engineering*, volume 35, issue 6 (2009), pp. 756–779. DOI: 10.1109/TSE.2009.67. URL: <http://ieeexplore.ieee.org/document/5353439/>.
- [180] G. Mussbacher, D. Amyot, R. Breu, J.-M. Bruel, B. H. Cheng, P. Collet, B. Combe-male, R. B. France, R. Heldal, J. Hill, et al. "The relevance of model-driven engineering thirty years from now." In: *International Conference on Model Driven Engineering Languages and Systems - MODELS'14*. Springer. 2014, pp. 183–200.
- [181] B. Nagel, C. Gerth, J. Post, and G. Engels. "Kaos4SOA-Extending KAOS Models with Temporal and Logical Dependencies." In: *CAiSE Forum*. 2013, pp. 9–16.
- [182] R. Normann and R. Ramirez. "From value chain to value constellation: Designing interactive strategy." In: *Harvard business review* 71.4 (1993), pp. 65–77.
- [183] D. North. *Introducing BDD*. Online accessed: 22/Jun/2018. 2006. URL: <https://dannorth.net/introducing-bdd/>.
- [184] J. C. Nunnally and I. H. Bernstein. "Psychometric theory." In: (1978).
- [185] B. J. Oates and G. Capper. "Using systematic reviews and evidence-based software engineering with masters students." In: *EASE*. Vol. 9. 2009, pp. 20–21.
- [186] Obeo. *UML Designer Documentation*. Online accessed: 22/Jun/2018. 2018. URL: <http://www.uml designer.org/>.

- [187] Obeo Designer. *Obeo Designer website*. Online accessed: 27/Jun/2018. URL: <https://www.obeodesigner.com/en/>.
- [188] L. B. R. de Oliveira and E. Y. Nakagawa. "A Systematic Review on Service-Oriented Reference Models and Service-Oriented Reference Architectures." In: *Software Architecture* 6285.29 (2010), pp. 360–367. DOI: 10.1007/978-3-642-15114-9_29. URL: http://link.springer.com/10.1007/978-3-642-15114-9_29.
- [189] OMG. *Documents Associated With Service Oriented Architecture Modeling Language (SoaML), Version 1.0.1*. Online accessed: 22/Sep/2017. URL: <https://goo.gl/FHHeUK>.
- [190] OMG. *UML Profile For Modeling Quality Of Service And Fault Tolerance Characteristics And Mechanisms (QFTP)*. URL: <http://www.omg.org/spec/QFTP/>.
- [191] A Osterwalder. "The Business Model Ontology-a proposition in a design science approach (2004)." Doctoral dissertation. Universite de Lausanne, 2004.
- [192] A. Osterwalder, Y. Pigneur, and C. L. Tucci. "Clarifying business models: Origins, present, and future of the concept." In: *Communications of the association for Information Systems* 16.1 (2005), p. 1.
- [193] M. Pai, M. McCulloch, J. D. Gorman, N. Pai, W. Enanoria, G. Kennedy, P. Tharyan, and J. J. Colford. "Systematic reviews and meta-analyses: an illustrated, step-by-step guide." In: *The National medical journal of India* 17.2 (2004), pp. 86–95.
- [194] R. Pandey. "Architectural description languages (ADLs) vs UML: a review." In: *ACM SIGSOFT Software Engineering Notes* 35.3 (2010), pp. 1–5.
- [195] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. "Service-oriented computing: State of the art and research challenges." In: *Computer* 40.11 (2007).
- [196] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. "Service-oriented computing: a research roadmap." In: *International Journal of Cooperative Information Systems* 17.02 (2008), pp. 223–255.
- [197] Papers. *Papers: Your personal library of research*. Online accessed: 22/Sep/2017. 2017. URL: <http://www.papersapp.com>.
- [198] A. G. Pateli and G. M. Giaglis. "A research framework for analysing eBusiness models." In: *European journal of information systems* 13.4 (2004), pp. 302–314.
- [199] D. J. Paulish and L. Bass. *Architecture-centric software project management: A practical guide*. Addison-Wesley Longman Publishing Co., Inc., 2001.
- [200] J. Pearl. "Heuristics: intelligent search strategies for computer problem solving." In: (1984).
- [201] D. E. Perry and A. L. Wolf. "Foundations for the study of software architecture." In: *ACM SIGSOFT Software engineering notes* 17.4 (1992), pp. 40–52.

-
- [202] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson. "Systematic Mapping Studies in Software Engineering." In: *EASE*. Vol. 8. 2008, pp. 68–77.
- [203] K. Petersen, S. Vakkalanka, and L. Kuzniarz. "Guidelines for conducting systematic mapping studies in software engineering: An update." In: *Information and Software Technology* 64 (2015), pp. 1–18.
- [204] P. Petrov, R. L. Nord, and U. Buy. "Probabilistic Macro-Architectural Decision Framework." In: *Proceedings of the 2014 European Conference on Software Architecture Workshops*. ACM. 2014, p. 27.
- [205] M. Petticrew and H. Roberts. *Systematic reviews in the social sciences: A practical guide*. John Wiley & Sons, 2008.
- [206] V. Pijpers, P. De Leenheer, J. Gordijn, and H. Akkermans. "Using conceptual models to explore business-ICT alignment in networked value constellations." In: *Requirements Engineering* 17.3 (2012), pp. 203–226.
- [207] M. E. Porter and V. E. Millar. *How information gives you competitive advantage*. Harvard Business Review Cambridge, MA, 1985.
- [208] C. U. Press. *Cambridge Dictionary*. Online accessed: 28/Jun/2018. 2018. URL: <https://dictionary.cambridge.org/pt/dicionario/ingles/framework>.
- [209] "Proceedings Working IEEE/IFIP Conference on Software Architecture." In: *Proceedings Working IEEE/IFIP Conference on Software Architecture*. 2001, pp. iii–. DOI: [10.1109/WICSA.2001.948397](https://doi.org/10.1109/WICSA.2001.948397).
- [210] J. Putman. *Architecting with RM-ODP*. Prentice-Hall Professional, 2001. ISBN: 9780130191168.
- [211] H. Raju. "The New Age of Innovation: Driving Co-Created Value through Global Networks." In: *Adarsh Journal of Management Research* 1.1 (2008), pp. 77–78.
- [212] A. Rasiwasia. "Meta Modeling for Business Model Design: Designing a Meta model for E3 value model based on MOF." Master's thesis. Royal Institute of Technology, 2013.
- [213] RespectIT. *A KAOS Tutorial*. v1.0. Oct. 2007.
- [214] S. Robertson and J. Robertson. *Mastering the requirements process: Getting requirements right*. Addison-wesley, 2012.
- [215] R. Rosenthal. *Meta-Analytic Procedures for Social Research*. Sage Publications, volume 6, Applied Social Research Methods Series, 1991.
- [216] K. J. Rothman, S. Greenland, and T. L. Lash. *Meta-Analysis*. Page 652 in *Modern epidemiology*. Lippincott Williams & Wilkins, 2008.
- [217] J. Rumbaugh, I. Jacobson, and G. Booch. *The unified modeling language reference manual*. Pearson Higher Education, 2004.

- [218] D. L. Sackett. *Evidence-based Medicine How to practice and teach EBM*. WB Saunders Company, 1997.
- [219] D. L. Sackett, W. M. Rosenberg, J. M. Gray, R. B. Haynes, and W. S. Richardson. *Evidence based medicine: what it is and what it isn't*. 1996.
- [220] J. Savolainen and V. Myllarniemi. "Layered architecture revisited—Comparison of research and practice." In: *Software Architecture, 2009 & European Conference on Software Architecture. WICSA/ECSA 2009. Joint Working IEEE/IFIP Conference on*. IEEE. 2009, pp. 317–320.
- [221] G. Scanniello and U. Erra. "Distributed modeling of use case diagrams with a method based on think-pair-square: Results from two controlled experiments." In: *Journal of Visual Languages & Computing, volume 25, issue 4* (2014), pp. 494–517. doi: [10.1016/j.jvlc.2014.03.002](https://doi.org/10.1016/j.jvlc.2014.03.002).
- [222] D. C. Schmidt. "Guest Editor's Introduction: Model-Driven Engineering." In: *IEEE Computer* 39.2 (2006).
- [223] S. Shaphiro and M. Wilk. "An analysis of variance test for normality." In: *Biometrika* 52.3 (1965), pp. 591–611.
- [224] M. Shaw. "Toward higher-level abstractions for software systems." In: *Data & Knowledge Engineering* 5.2 (1990), pp. 119–128.
- [225] A. P. F. Silva. "Uma abordagem ágil para transformar modelos cognitivos em modelos comportamentais e de domínio." Master's thesis. New University of Lisbon, 2014.
- [226] Y. Singh and M. Sood. "Model driven architecture: A perspective." In: *Advance Computing Conference, 2009. IACC 2009. IEEE International*. IEEE. 2009, pp. 1644–1652.
- [227] C. Siobhan and B. Elisa. *Aspect-Oriented Analysis and Design: The Theme Approach*. 2005.
- [228] V. Siochos and C. Papatheodorou. "Developing a formal model for mind maps." In: *First Workshop on Digital Information Management, Greece* (2011).
- [229] Slashdot Media. *Freemind tool*. Online accessed: 22/Jun/2018. 2018. URL: http://freemind.sourceforge.net/wiki/index.php/Main_Page.
- [230] K. Smolander. "What is included in software architecture? A case study in three software organizations." In: *IEEE International Conference and Workshop on the Engineering of Computer-Based Systems*. IEEE. 2002.
- [231] Software Engineering Institute. *What is your definition of software architecture?* <https://goo.gl/PBCLWR>. 2010.
- [232] I. Solheim and K. Stølen. "Technology research explained." In: *Technical Report SITEF A313. SINTEF ICT*, 2007.

-
- [233] C. Solis and X. Wang. "A study of the characteristics of behaviour driven development." In: *Software Engineering and Advanced Applications (SEAA), 2011 37th EUROMICRO Conference on*. IEEE. 2011, pp. 383–387.
- [234] A Sølvsberg and C. Kung. "Activity modelling and behaviour modelling." In: *Information Systems Design Methodologies: Improving the Practice* (1986).
- [235] I. Sommerville. *Software engineering*. 9th edition. Pearson, 2010.
- [236] I. Sommerville and G. Kotonya. *Requirements Engineering: Processes and Techniques*. John Wiley & Sons, Inc., 1998.
- [237] E. Souza and A. Moreira. "Deriving Services using KAOS Models." In: *33rd ACM/SIGAPP Symposium On Applied Computing (SAC2018), Pau, France*. 2018.
- [238] E. Souza, S. Abrahão, A. Moreira, J. Araújo, and E. Insfran. "Comparing Value-Driven Methods: an Experiment Design." In: *HuFaMo@ MoDELS, Saint Malo, France*. 2016, pp. 19–26.
- [239] E. Souza, A. Moreira, and J. Araújo. "Aligning Business Models with Requirements Models." In: *European, Mediterranean, and Middle Eastern Conference on Information Systems, Coimbra, Portugal*. Springer. 2017, pp. 545–558.
- [240] E. Souza, A. Moreira, and C. De Faveri. "An approach to align business and IT perspectives during the SOA services identification." In: *17th International Conference on Computational Science and Its Applications (ICCSA), Trieste, Italy*. IEEE. 2017, pp. 1–7.
- [241] E. Souza, S. Abrahao, A. Moreira, E. Insfran, and J. Araújo. "Evaluating the efficacy of value-driven methods: a controlled experiment." In: *26th International Conference on Information Systems Development (ISD2017), Larnaca, Cyprus*. 2017.
- [242] E. Souza, S. Abrahao, A. Moreira, J. Araújo, and E. Insfran. *Value-Driven Development Method Survey Instrument*. Available on <https://goo.gl/forms/6KJA3zXyUFx3iHa62>. 2017.
- [243] E. Souza, A. Moreira, J. Araújo, S. Abrahão, E. Insfran, and D. S. da Silveira. "Comparing business value modeling methods: A family of experiments." In: *Information and Software Technology* (2018). ISSN: 0950-5849. DOI: <https://doi.org/10.1016/j.infsof.2018.08.001>. URL: <http://www.sciencedirect.com/science/article/pii/S0950584918301629>.
- [244] E. Souza, A. Moreira, and F. Wanderley. "Towards an Agile Reference Architecture Method for Information Systems." In: *Proceedings of the 51st Hawaii International Conference on System Sciences, Hawaii, USA*. 2018.

- [245] E. Souza, A. Moreira, and M. goulao. “Deriving Architectural Models from Requirements Specifications: a Systematic Mapping Study.” In: *Information and Software Technology* (2019). issn: 0950-5849. doi: <https://doi.org/10.1016/j.infsof.2019.01.004>. url: <https://www.sciencedirect.com/science/article/abs/pii/S0950584919300035>.
- [246] Springer. *SpringerLink Digital Library*. Online accessed: 22/Sep/2017. 2017. url: <https://link.springer.com>.
- [247] Standish Group. *Standish Group Website*. <http://www.standishgroup.com>. 2017.
- [248] R Studio. “RStudio: integrated development environment for R.” In: *RStudio Inc, Boston, Massachusetts* (2012).
- [249] A. J. Sutton, K. R. Abrams, and D. R. Jones. “An illustrated guide to the methods of meta-analysis.” In: *Journal of evaluation in clinical practice* 7.2 (2001), pp. 135–148.
- [250] D. Tapscott. *Digital capital: Harnessing the power of business webs*. Harvard Business School Press, 2000.
- [251] B. Team. *Business motivation model (bmm) specification*. Tech. rep. Needham, Massachusetts. Technical Report dtc/06–08–03, Object Management Group, 2006.
- [252] B. Tekinerdogan. “ASAAM: Aspectual software architecture analysis method.” In: *Software Architecture, 2004. WICSA 2004. Proceedings. Fourth Working IEEE/IFIP Conference on*. IEEE. 2004, pp. 5–14.
- [253] Thales. *Thales Group website*. Online accessed: 27/Jun/2018. url: <https://www.thalesgroup.com/en>.
- [254] G. H. Travassos, D. Gurov, and E. Amaral. *Introdução à engenharia de software experimental*. UFRJ, 2002.
- [255] J. Tyree and A. Akerman. “Architecture decisions: Demystifying architecture.” In: *IEEE software* 22.2 (2005), pp. 19–27.
- [256] J. S. Van Der Ven, A. Jansen, J. Nijhuis, and J. Bosch. “Design decisions: The bridge between rationale and architecture.” In: *Rationale management in software engineering* 16 (2006).
- [257] A. Van Lamsweerde. “Goal-oriented requirements engineering: A guided tour.” In: *Proceedings fifth ieee international symposium on requirements engineering*. IEEE. 2001, pp. 249–262.
- [258] R. Van Solingen. “Measuring the ROI of software process improvement.” In: *IEEE software* 21.3 (2004), pp. 32–38.
- [259] F. Wanderley and J. Araujo. “Generating goal-oriented models from creative requirements using model driven engineering.” In: *Model-Driven Requirements Engineering (MoDRE), 2013 International Workshop on*. IEEE. 2013, pp. 1–9.

-
- [260] F. Wanderley and D. S. da Silveria. "A framework to diminish the gap between the business specialist and the software designer." In: *Quality of Information and Communications Technology (QUATIC), 2012 Eighth International Conference on the*. IEEE. 2012, pp. 199–204.
- [261] R. Weber and Y. Zhang. "An analytical evaluation of NIAM's grammar for conceptual schema diagrams." In: *Information Systems Journal* 6.2 (1996), pp. 147–170.
- [262] H. Weigand, P. Johannesson, B. Andersson, and M. Bergholtz. "Value-based service modeling and design: Toward a unified view of services." In: *Advanced Information Systems Engineering*. Springer. 2009, pp. 410–424.
- [263] S. A. White. "Introduction to BPMN." In: *IBM Cooperation 2.0* (2004), p. 0.
- [264] S. A. White. "Process modeling notations and workflow patterns." In: *Workflow handbook 2004* (2004), pp. 265–294.
- [265] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in software engineering*. Springer-Verlag Berlin Heidelberg, 2012. ISBN: 978-3-642-29043-5. DOI: [10.1007/978-3-642-29044-2](https://doi.org/10.1007/978-3-642-29044-2). URL: <http://link.springer.com/10.1007/978-3-642-29044-2>.
- [266] R. Wojcik, F. Bachmann, L. Bass, P. Clements, P. Merson, R. Nord, and B. Wood. *Attribute-driven design (ADD), version 2.0*. Tech. rep. Carnegie-Mellon University Pittsburgh PA Software Engineering Institute, 2006.
- [267] C. Yang, P. Liang, and P. Avgeriou. "A systematic mapping study on the combination of software architecture and agile development." In: *Journal of Systems and Software* 111 (2016), pp. 157–184.
- [268] E. Yu, P. Giorgini, N. Maiden, and J. Mylopoulos. *Social modeling for requirements engineering*. Mit Press, 2011.
- [269] H. Zhang, M. A. Babar, and P. Tell. "Identifying relevant studies in software engineering." In: *Information and Software Technology* 53.6 (2011), pp. 625–637.
- [270] L. Zhu and I. Gorton. "Uml profiles for design decisions and non-functional requirements." In: *Proceedings of the Second Workshop on Sharing and Reusing Architectural Knowledge Architecture, Rationale, and Design intent*. IEEE Computer Society. 2007, p. 8.
- [271] C. Zott, R. Amit, and L. Massa. "The business model: recent developments and future research." In: *Journal of management* 37.4 (2011), pp. 1019–1042.

A Value-Driven Framework for Software Architecture

Eric Souza





Eric Rocha de Souza

BSc in Information Systems
Specialist in Software Engineering
MSc in Computer Engineering

A Value-Driven Framework for Software Architecture

Thesis submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy in
Computer Science

September, 2019



FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA



Eric Rocha de Souza

BSc in Information Systems
Specialist in Software Engineering
MSc in Computer Engineering

**A Value-Driven Framework
for Software Architecture**

Thesis submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy in
Computer Science

September, 2019

Copyright © Eric Rocha de Souza, Faculty of Sciences and Technology, NOVA University Lisbon.

The Faculty of Sciences and Technology and the NOVA University Lisbon have the right, perpetual and without geographical boundaries, to file and publish this dissertation through printed copies reproduced on paper or on digital form, or by any other means known or that may be invented, and to disseminate through scientific repositories and admit its copying and distribution for non-commercial, educational or research purposes, as long as credit is given to the author and editor.



FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA